ABSTRACT

Title of dissertation: DEEP NEURAL NETWORKS FOR RADIO FREQUENCY FINGERPRINTING

Kevin Merchant
Doctor of Philosophy, 2019

Dissertation directed by: Professor Rama Chellappa
Department of Electrical and Computer Engineering

As the Internet of Things (IoT) continues to expand, there is a growing necessity for improved techniques to authenticate the identity of wireless transmitters to prevent unauthorized network access. In this dissertation, we develop a series of physical-layer authentication, or radio frequency (RF) fingerprinting, techniques which utilize methods from deep learning to train convolutional and recurrent neural network models to verify the identity of wireless transmitters which meet the IEEE 802.15.4 standard.

First, we develop a technique which utilizes a convolutional neural network (CNN) to identify or verify the identity of a transmitter from which a time-domain complex baseband signal was recorded. This technique relies on an extensive preprocessing sequence to remove sources of potential bias and trivial features from the received waveforms, and derives an estimated error signal from each recording from which the CNN learns discriminatory features. We demonstrate the effectiveness of

the technique on a set of seven off-the-shelf ZigBee devices recorded outside in an urban environment, as well as in a laboratory environment with artificial noise over a wide-range of signal-to-noise ratios (SNRs).

Next, we train a series of models which utilize both convolutional and recurrent elements to improve the performance of the previous technique in the presence of high levels of noise and expand the evaluation to a larger set of twenty-five devices. We evaluate several realistic scenarios, including the performance in typical multipath environments and the ability to correctly reject previously unseen devices. In order to justify the proposed pre-processing sequence, we present experimental results that demonstrate weaknesses in fingerprint verification classifiers in which frequency synchronization is not performed. Finally, we present a simple technique to reduce the amount of memory required for a collection of fingerprint models by up to 95% without loss of performance.

To further enhance the security of the trained fingerprint models, we propose a generative adversarial network (GAN) architecture and training procedure to provide additional training examples for the classifiers. We show that fingerprint classifiers that are trained exclusively on real devices cannot reliably reject GAN-generated signals. Furthermore, we illustrate that augmenting the training process of the fingerprint models with GAN-generated signals reduces this vulnerability, even if the GAN used for training and inference are different.

Finally, we assess the practicality of transferring an RF fingerprint model from one receiver to another. Experimentally, we demonstrate significant degradation in classification performance when a fingerprint model is learned using signals recorded

on one receiver and evaluated using signals recorded on another receiver. First, we show that generalization may be improved by including multiple receivers in the training process. Then, we develop a calibration procedure whereby models learned on a single receiver can be transferred without alteration to another receiver by learning a transformation function, implemented as a residual neural network, to model the variations between the two receivers. We perform several experiments with ten commercial receivers to confirm the effectiveness of the technique under realistic constraints.

DEEP NEURAL NETWORKS FOR RADIO FREQUENCY FINGERPRINTING

by

Kevin Merchant

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Advisory Committee:
Professor Rama Chellappa, Chair/Advisor
Professor K.J. Ray Liu
Professor Sennur Ulukus
Doctor George Stantchev
Professor Ashok Agrawala, Dean's Representative

Dedication

To my mom, for her love and support.

Acknowledgments

First, I would like to thank my advisor, Prof. Rama Chellappa, for his assistance and motivation throughout the duration of my Ph.D. His guidance through the entire process is greatly appreciated.

I would like to thank my advisory committee, Prof. K.J. Ray Liu, Prof. Sennur Ulukus, Dr. George Stantchev, and Prof. Ashok Agrawala, for their review, feedback, and help with this dissertation.

I would like to thank my collaborators throughout the years at the US Naval Research Laboratory, particularly the employees of Code 5730. Specifically, I would like to thank Bryan Nousain, Shauna Revay, and Mike Spath. Their assistance has been invaluable, and they have greatly contributed to my development as a researcher.

I would like to thank Prof. Ankur Srivastava for motivating me to pursue a Ph.D. in the first place.

I would like to thank my friends for their support throughout this process. In particular, I appreciate the friendship and support of Mike Zuzak, Manal Assad, Michael Titus, and Danny Kim. I would also like to thank my girlfriend, Alana Heifetz, for her support and encouragement.

Finally, I would like to thank my family for their unwavering support, motivation, and assistance throughout my life and education. Without them, this would have never been possible.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

1D	One-Dimensional
2D	Two-Dimensional
ACARS	Aircraft Communications Addressing and Reporting System
ADS-B	Automatic Dependent Surveillance - Broadcast
AIR-T	Artificial Intelligence Radio Transceiver
AUC	Area Under the Curve
AWGN	Additive White Gaussian Noise
CNN	Convolutional Neural Network
CSI	Channel State Information
DAC	Digital-to-Analog Converter
DAGAN	Data Augmentation Generative Adversarial Network
dB	Decibel
dBm	Decibel-Milliwatts
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DC	Direct Current
DFT	Discrete Fourier Transform
DNL	Differential Nonlinearity
DQPSK	Differential Quadrature Phase Shift Keying
DSSS	Direct-Sequence Spread Spectrum
DUT	Device Under Test
DWFP	Dynamic Wavelet Fingerprinting
ELU	Exponential Linear Unit
EVM	Error Vector Magnitude
FIR	Finite Impulse Response
FPR	False Positive Rate
FSK	Frequency Shift Keying
GAN	Generative Adversarial Network
GHz	Gigahertz
GPU	Graphics Processing Unit
HDF5	Hierarchical Data Format 5
HHT	Hilbert-Huang Transform
Hz	Hertz

IEEE	Institute of Electrical and Electronics Engineers
INL	Integral Nonlinearity
IoT	Internet of Things
I/Q	In-phase and Quadrature Components
ISM	Industrial, Scientific, and Medical
Kbps	Kilobits per second
KL	Kullback-Leibler
kNN	k-Nearest Neighbors
LDA	Linear Discriminant Analysis
LSTM	Long Short-Term Memory
LTI	Linear Time-Invariant
MAC	Media Access Control
MDA	Multiple Discriminant Analysis
MHz	Megahertz
MLE	Maximum Likelihood Estimation
MLP	Multilayer Perceptron
OFDM	Orthogonal Frequency-Division Multiplexing
O-QPSK	Offset Quadrature Phase Shift Keying
PCA	Principal Component Analysis
PIN	Personal Identification Number
PNN	Probabilistic Neural Network
P-R	Precision-Recall
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
ReLU	Rectified Linear Unit
ResNet	Residual Neural Network
RF	Radio Frequency
RFID	Radio Frequency Identification
RMS	Root Mean Square
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
SD	Standard Deviation
SDA	Subclass Discriminant Analysis

SFD	Start Frame Delimiter
SHR	Synchronization Header
SNR	Signal-to-Noise Ratio
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TDMA	Time-Division Multiple Access
TPR	True Positive Rate
USRP	Universal Software Radio Peripheral
WPAN	Wireless Personal Area Network
WPD	Wavelet Packet Decomposition

Chapter 1: Introduction

1.1 Motivation

Recently, the availability of high-speed graphics processing units (GPUs), large datasets, and open-source software libraries that implement machine learning algorithms has led to a surge of research in deep neural networks, largely concentrated within the domains of computer vision and natural language processing. Notably, a number of deep learning-based techniques have been developed to address common privacy and security concerns, such as user authentication, by learning biometric features. These features, which may include fingerprints, irises, and faces, are specific to the user and are much more difficult to replicate than a compromised personal identification number (PIN) or password, which can simply be entered by an unauthorized user to gain access to a system.

Despite the overwhelming amount of research being done within these domains, relatively little work has been published that applies deep learning algorithms to another critical area of security: wireless communications. Currently, authentication protocols in wireless security rely on the secrecy of keys comprised of a sequence of bits. Unlike biometric data, which is difficult to reliably spoof, a bit-level key, if compromised, could be reproduced exactly and exploited to gain unauthorized

access to or privileges within a network while avoiding detection.

Analogous to the transition toward the analysis of human biometric data for security purposes, this dissertation seeks to apply deep learning algorithms to perform radio frequency (RF) fingerprinting, which is the use of physical-layer characteristics in a transmitted waveform to authenticate the identity of a transmitting device. Specifically, our work aims to train deep neural networks on received communications signals to identify the patterns which are unique to each transmitter and create a classifier to distinguish a target transmitter from all others.

Our work targets devices which implement the IEEE 802.15.4 protocol [1], a standard which defines the physical layer and media access control (MAC) layer for low data-rate wireless personal area networks (WPANs), commonly implemented in devices which comprise the Internet of Things (IoT). These devices, such as those that implement the ZigBee standard [2], tend to be low cost and manufactured by a wide variety of companies. As such, we expect and demonstrate that the variability between devices is sufficiently high to distinguish them, even those of the exact same model.

1.2 Problem Statement

Our goal is to be able to verify whether or not the identity of a transmitting device matches the identity that it claims using only the characteristics present in the transmitted waveform, with no other prior knowledge except the communications standard, in this case IEEE 802.15.4. In addition, we aim to assess the

applicability of such techniques in several real-world scenarios, such as operating in noisy multipath channels and sharing fingerprint models amongst a set of receivers.

In doing so, we permit the use of a two-factor authentication scheme whereby a device may verify its identity through use of a secret bit-level key in addition to its RF fingerprint. To overcome this scheme and effectively spoof a trusted device, an adversary would have to compromise the secret bit-level key and mimic the physical waveform that would be produced by the trusted device to a high degree of accuracy. As we will show, even devices of the same model have sufficient variation in their waveforms to distinguish them. An attack would thus require a sophisticated approach to learn and transmit a waveform that is indistinguishable from the trusted device.

In addition, such techniques could allow for a receiver to detect if the authentication keys it is using have been compromised and respond accordingly. As soon as a packet is received that passes the bit-level authentication, but fails the RF fingerprint verification beyond a certain threshold, the receiver can infer that its keys have been compromised and should no longer be trusted. Instead, the receiver can ignore all transmissions from the now-compromised address until a new pair of keys has been generated. If the RF fingerprinting algorithm has a low enough failure rate, new keys could even be generated and trusted on the fly by accepting them only if they were generated by a device that passes the RF fingerprint validation. In practice, this would require a trade-off between false positive and true positive rates for the classifier that would need to be set carefully depending on the needs of the application.

Our work presents two classification problems:

1. The *verification problem* seeks to confirm whether or not a given transmission originates from a particular device. In a practical setting, a device may be verified against the fingerprint model corresponding to the source address claimed in the transmission. This is a binary one-vs-all classification problem, and performance may be evaluated using receiver operating characteristic (ROC) and precision-recall (P-R) curves. For verification, we train a single neural network model per device to confirm the fingerprint of that device against all others.
2. The *identification problem* seeks to select which device among those in the training set produced a given transmission. This is a multi-class problem and performance may be evaluated using confusion matrices and correct classification rates. For identification, we develop a single neural network for each experiment to assign transmissions to devices in the training set. This problem is less applicable to a practical cybersecurity context since it is unlikely that a particular untrusted device is seen during training, but better demonstrates the degree of uniqueness within the population of devices.

In order to solve the verification problem, each target device requires a neural network that distinguishes that particular device from all others. In a practical context, it is cumbersome to manually tune the architecture and hyperparameters for each target device. Instead, we focus on developing a single network model and training procedure that can be effectively implemented for each device.

Conversely, the identification problem is a classic example of a multi-class classification problem, for which one approach is to use a single neural network to assign a recorded signal to any one of the devices in the training set.

We define three classes of adversaries who may attempt to claim the identity of an authorized device within a network to fool a receiver into unknowingly performing an unauthorized task:

- *Weak adversaries* simply transmit with another device, possibly of the exact same model, and modify the data bits that are transmitted to claim the identity of a trusted device. Such an adversary has compromised bit-level credentials, but makes no sophisticated attempts to replicate the RF fingerprint.
- *Intermediate adversaries* perform the same attack as a weak adversary, but also change high-level characteristics of their transmitted waveform that are assumed to be uniform throughout the duration of a packet, such as the exact center frequency, power level, or transmit location, in an effort to appear more similar to a trusted device.
- *Strong adversaries* strengthen their attack further by performing fine-grained adjustments to their transmitted waveform on a sample-by-sample basis to appear as similar as possible to an authorized device. Such an adversary may have developed an accurate model of the RF front-end of the target device by listening to its transmissions.

1.3 Contributions

The first two techniques we present attempt to detect weak and intermediate adversaries. Our first method develops a data pre-processing procedure to remove certain features from recorded wireless signals that may permit trivial classification, and then trains a convolutional neural network (CNN) to distinguish complex baseband error signals from different devices [3]. We define the complex baseband error signal as the additional signal beyond the ideal waveform for a particular bit sequence. Since CNNs operate on fixed-size data, we develop a technique to split received signals into fixed-size windows, classify the signal contents of each window using the CNN, then combine the outputs from all windows within each transmission to form a final classification. We demonstrate the performance of this method on data from a set of seven commercially-available devices recorded in a laboratory environment with simulated additive noise, as well as on data collected outdoors in an urban environment.

Our second method removes the need to manually split and recombine signals through use of a recurrent neural network (RNN) structure known as a long short-term memory (LSTM) [4] which operates over the time dimension [5]. This method also improves upon the pre-processing sequence and demonstrates better performance at lower signal-to-noise ratios (SNRs) than previously. Additionally, we develop a simulated multipath model based on published measurements from ZigBee transmitters and demonstrate that the sort of weak multipath effects that are likely to be present in a WPAN minimally impact classifier performance, pro-

vided that they are accounted for during the training process. This work also shows that the verification networks generalize well to *withheld* devices, those that were not seen during training, by correctly classifying them as an unauthorized device. Another experiment emphasizes the importance of frequency synchronization during pre-processing by showing that a network trained on unsynchronized transmissions performs well in the presence of weak adversaries, but is vulnerable to an intermediate adversary that simply frequency-shifts their transmissions. Finally, we show that a reduced memory model, in which all verification networks share 95% of their weights, is possible without adversely impacting performance.

Next, we modify the generative adversarial network (GAN) formulation introduced by Goodfellow *et al.* [6] for use with the RF fingerprinting problem [7]. In doing so, we are able to train a model of each transmitter to produce an unlimited amount of data for training the verification models. These GAN models represent the strong adversary which has precise sample-by-sample control over their output waveform. We show experimentally that the GAN models are able to successfully fool verification networks which were trained exclusively on real devices. Finally, we demonstrate that using a GAN to augment the training dataset for the verification networks can successfully defend against this vulnerability with high probability, even if the GAN architecture and dataset used for training are different than those used for evaluation.

Finally, we analyze the impact of the receiver on classifier performance [8]. The prior work has assumed that the same receiver is used for training and testing and that the verification models will never need to be adapted for use with another

receiver. This work investigates that assumption on a collection of ten commercially-available receivers. First, we demonstrate that training a model on one receiver and then testing it on another results in significantly degraded verification performance. We propose two methods to reduce this effect. The first simply requires training the verification networks on data recorded using multiple receivers. We show that this provides significant improvement, particularly at high SNRs. Next, we introduce a calibration procedure whereby a model that has already been trained using a single receiver may be deployed to new receivers by learning a residual neural network (ResNet) [9] transformation for each receiver to adjust the complex baseband signal input prior to classification by the verification network. We demonstrate that this approach reduces the error when switching receivers by 32.09% to 62.99%, depending on the SNR. We then present the results of several experiments using the proposed transform method, including training with several *virtual receivers* that are simulated using the learned transformations, and replacing the original training receiver with another that is augmented by its own transformation network.

The remainder of this dissertation is organized as follows: Chapter 2 summarizes related work that seeks to solve the RF fingerprinting problem. Chapter 3 provides background information on the IEEE 802.15.4 standard and the sort of transmitter imperfections that make it possible to perform fingerprinting. Chapter 4 introduces a CNN technique for solving both the identification and verification problems. Chapter 5 improves upon the previous technique with enhanced preprocessing and the addition of RNN components and provides further analysis of the performance in realistic scenarios. Chapter 6 introduces a GAN approach to

improving the security of trained fingerprint verification models. Chapter 7 analyzes the ability a fingerprint to be used with a receiver other than the one used for training and develops a ResNet technique for adapting signals from a new receiver for use with the original model. Finally, Chapter 8 provides concluding remarks and possible future directions for the work.

Chapter 2: Related Work

A large amount of prior work has been conducted on the RF fingerprinting problem, much of it focused on techniques which require domain-specific expert knowledge to carefully develop hand-crafted feature vectors on which to classify the transmitters using standard approaches from machine learning. We review these techniques in Section 2.1.

Other authors have taken a neural network approach for feature extraction and/or classification. Some of these models follow a hybrid approach, combining hand-crafted feature vectors with neural network classifiers. In contrast, our methods follow a more data-driven approach and utilize neural network models to perform both feature extraction and classification. Recently, there has been great interest in applying neural network models to the RF fingerprinting problem, due at least in part to its inclusion in the Defense Advanced Research Projects Agency (DARPA) Radio Frequency Machine Learning Systems (RFMLS) program [10]. Techniques based at least in part on neural network models are reviewed in Section 2.2.

Until recently, the application of neural network models to the RF domain has been relatively limited. However, the advent of deep learning techniques, which have demonstrated groundbreaking performance in a wide-array of classification

problems, particularly in the domains of image, video, and audio classification [11–13], has motivated a number of applications of neural network models to other problems in the RF domain. This work, though not directly aimed toward solving the RF fingerprinting problem, provides useful insights and is summarized in Section 2.3.

2.1 RF Fingerprint Classifiers Based on Hand-Crafted Feature Vectors

A commonly used feature extraction technique for WPAN devices such as ZigBee and Z-Wave is to divide a fixed header portion of the signal, such as the preamble or synchronization header (SHR), into a number of equal-sized regions and compute statistics such as variance, skewness, and kurtosis on the instantaneous amplitude, phase, or frequency signal in each region [14–17]. Classification algorithms based on multiple discriminant analysis (MDA), maximum likelihood estimation (MLE), and random forests can then be trained to learn a distribution of feature vectors from the data in order to distinguish transmissions from different devices. Previous work has shown that phase features are the most useful for distinguishing ZigBee devices [16] which implement the IEEE 802.15.4 protocol.

Alternatively, methods have been proposed which seek to classify based on patterns in time-frequency representations of measured signals. Yuan *et al.* developed a classifier which selects features from the Hilbert-Huang transform (HHT) of the transient portion of eight emitters, then uses principal component analysis

(PCA) to reduce the dimensionality of the feature space before training a support vector machine (SVM) classifier [18]. Similarly, Ali *et al.* distinguish Bluetooth transmissions from 20 cell phones by creating a set of hand-crafted features directly from the transient portion of the signal, including features from the HHT of the transient, then performing classification using decision trees, SVMs, and linear discriminant analysis (LDA) [19]. Building off of a number of techniques which utilize only the rising transient of the signals for fingerprinting, Yang *et al.* show that adding features computed from the falling transient and a synchronization portion of the transmission can boost the accuracy of the classifier beyond that achieved using only the rising transient [20].

Bertoncini *et al.* created a feature vector by combining features from three different methods: dynamic wavelet fingerprinting (DWFP), wavelet packet decomposition (WPD), and higher order statistics of the recorded signal [21]. They then demonstrated that a SVM classifier can be trained to reliably distinguish transmissions from 146 radio frequency identification (RFID) tags recorded in a high SNR environment. Xie *et al.* likewise combine wavelet transform techniques for multiresolution analysis with a SVM classifier to perform fingerprinting, but they additionally apply a coherent integration technique, typically used in radar systems, as a denoising algorithm prior to feature extraction [22]. They show that this denoising technique significantly enhances the performance of the classifier on low SNR signals. Similarly, Zhou *et al.* demonstrated that averaging the recorded time-series over a repeated symbol sequence improves the ability of the fingerprint classifier to generalize to signals recorded over a longer span of time [23]. In their

case, they showed that the averaging technique allowed fingerprints generated from signals recorded 18 months prior to successfully be used to classify a population of 50 ZigBee devices with 99.71% accuracy using a Bayesian approach and modeling the signals with multivariate Gaussian distributions.

Huang and Zheng demonstrated an ability to distinguish seven time-division multiple access (TDMA) satellite terminals with quadrature phase shift keying (QPSK) by measuring the error in the constellation of the steady state portion of the received signal [24]. Their method then uses LDA and subclass discriminant analysis (SDA) to separate the constellation error signals into each of the seven classes. Ali *et al.* showed that statistics of the error vector can be used to distinguish real base stations from fake base stations due to the quality of components and digital predistortion techniques used in real base stations to correct power amplifier nonlinearity [25]. They demonstrate that by measuring the kurtosis of the distribution of errors, they can distinguish the error due to nonlinearity, which is much higher in the fake base stations generated using software defined radios, from random noise.

Brik *et al.* demonstrated a technique that trains SVM and k-nearest neighbor (kNN) classifiers to identify IEEE 802.11 transmitters by using high-level statistics of their modulation error as features [26]. Their feature vector includes the following five metrics: frequency error, phase error, magnitude error, distance between in-phase/quadrature (I/Q) origins of the ideal and measured signals, and correlation of I/Q values between the ideal and measured preamble sequences. Pan *et al.* developed a technique specifically for TDMA signals to assess whether adjacent

time slots within a TDMA frame belong to the same transmitter by measuring the carrier phase continuity between adjacent bursts and used their technique to enhance predictions given by the algorithm developed by Brik *et al.* [27]. Candore *et al.* used a similar set of modulation error statistics as features, but built a set of 14 weak single-variable classifiers by learning the approximate probability distribution of each feature for each device, then assigning a transmission to the class with the highest likelihood for the observed feature [28]. Finally, a strong classifier was demonstrated by weighting the output of the weak classifiers according to weights derived by the performance on a validation set. Experimental results showed strong performance at identifying six radios using differential QPSK (DQPSK) modulation.

Peng *et al.* showed that cluster centers in an image called a "differential constellation trace figure," which is derived from the signal, could be used along with frequency offset, modulation offset, and I/Q offset to build a lightweight classifier for use with ZigBee devices [29]. However, they also demonstrated an increase in the error rate by more than a factor of two when the technique is used in non-line-of-sight conditions and also when the test signals are recorded 18 months after the training signals using a different receiver compared to the baseline test.

Liu and Doherty showed that IEEE 802.11a/g devices can be distinguished by observing the spectral regrowth, a distortion caused by nonlinear power amplifiers [30]. By modeling the power amplifiers as memoryless nonlinear systems and using the learned complex power series coefficients for each device as a feature vector, they were able to show that three devices operating at a fixed power level are separable in two dimensions. Furthermore, they demonstrated that changing the power level

of each device moves the feature vector along a predictable trajectory, allowing the devices to be distinguished in three dimensions for variable power levels.

Carroll showed that embedding a recorded signal in a higher dimension using a time-delay embedding technique allows for the classification of wireless devices by device-specific linear and nonlinear behavior, which may be different across devices even of the same model due to manufacturing variability [31,32]. Classification of the embedded signal requires direct comparison with an embedded signal with known transmitter assignment. Unfortunately, when using a large training set, this requires a prohibitively large number of direct comparisons to classify a single signal. Yuan *et al.* developed a technique based on the embedding signal that does not require such direct comparisons [33]. Their method performs singular value decomposition (SVD) on the matrix representing the signal embedding, as well as on matrices measuring the angle and distance between neighboring points, and uses the singular values as a feature vector. Their method then uses PCA for dimensionality reduction and a SVM to classify signals from four IEEE 802.11b transmitters. They demonstrate 90%+ correct identification rates at 10 dB SNR and above.

Finally, Andrews *et al.* developed techniques for crowdsourcing measurements for RF fingerprinting, including ways to combine high-level decisions, features, and full signals recorded at the same time from several different receivers [34]. They examine possible benefits to this sort of setup, including the possibility of sub-Nyquist sampling at individual receivers and increased performance under certain conditions.

2.2 RF Fingerprint Classifiers Based on Neural Network Models

Willson, as well as Shieh and Lin, used neural network models acting on hand-crafted feature vectors to identify individual radar transmitters [35, 36]. However, the network input parameters, such as pulse width and pulse repetition interval, are specific to the radar domain and would not be relevant in a typical digital communications setting.

Üreten and Serinken trained a probabilistic neural network (PNN) to distinguish eight IEEE 802.11b WiFi cards using the transient amplitude signal of the transmissions, along with PCA for dimensionality reduction [37]. Similarly, Köse *et al.* developed a technique where a PNN classifies eight IEEE 802.11b WiFi cards by the energy spectrum coefficients from the transient turn-on signal [38]. These techniques are noteworthy as they combine an expert knowledge of distinguishable signal characteristics with a neural network which uses a data-driven approach to learn the most relevant features for discrimination. Our approach is different in that it learns features from the steady state portion of the transmission. This is useful as the steady state portion is typically received at a higher average power level and has a much longer duration than the transient. Consequently, their techniques required a sampling rate of 5 GHz to sufficiently sample the transient, while our techniques operate at a sample rate of 16 MHz. In addition, our methods utilize the complex baseband signal, rather than just the amplitude signal, allowing our technique to potentially learn additional features that are present in the signal phase, which may be more relevant for the types of devices we study [16].

More recently, a number of techniques for RF fingerprinting using neural network structures for feature extraction and, in most cases, classification have been published. However, most of these techniques operate exclusively on preamble or control information rather than randomly generated data payloads, and do not utilize a careful pre-processing sequence as is presented in this dissertation to limit the ability of the network to learn unreliable features, or those which can easily be spoofed.

Jafari *et al.* evaluated multilayer perceptron (MLP), CNN, and LSTM networks for classifying baseband signals from six ZigBee devices, finding that MLP and CNN structures greatly surpass the performance of the LSTM network [39]. Contrarily, Wu *et al.* trained a LSTM network to distinguish six universal software radio peripherals (USRPs) from their baseband signals with high performance down to -12 dB SNR [40]. Riyaz *et al.* demonstrated that a CNN architecture based on AlexNet [11] and operating on baseband signals outperforms SVM and logistic regression classifiers which use hand-crafted feature vectors [41]. In contrast to these simpler approaches, several authors have introduced additional techniques in an effort to enhance the ability to discern transmitters. Jiabao *et al.* showed that presenting the signals to a CNN architecture at multiple sample rates simultaneously can provide a slight boost to identification accuracy [42]. Additionally, Youssef *et al.* showed that sequentially training a series of MLPs and stacking them in a hierarchical structure, a process termed "multi-stage training," can boost performance over SVM, CNN, and single-MLP approaches on a set of 12 transmitters [43].

Several authors have transformed the time-domain signal into a 2D image, an

input format where neural networks have been extensively studied. Baldini *et al.* utilized a CNN structure operating on images produced using recurrence plots, a visual representation of the phase space trajectory of the signal [44]. They report increased identification accuracy compared to a 1D time-domain signal input at higher SNR, but worse performance at lower SNR. Pan *et al.* demonstrated that power amplifier nonlinearity specifically could be used as an identifying feature by differentiating five simulated transmitters with different polynomial amplifier nonlinearity characteristics [45]. Their approach used grayscale images produced by the HHT as input to a ResNet architecture for feature extraction and classification.

Contrary to most other approaches, which use strictly real-valued weights, Gopalakrishnan *et al.* investigated CNN approaches using complex-valued neural networks by separately computing derivatives for the real and imaginary parts of each weight during backpropagation and considering two different complex activation functions [46]. In doing so, they demonstrated successful RF fingerprint identification on sets of 100 WiFi devices and 100 automatic dependent surveillance-broadcast (ADS-B) devices, a technology used for tracking aircraft. Operating also on aircraft signals, Chen *et al.* presented a network architecture based on ResNets and Inception modules [47] to separately identify transmitters that implement ADS-B and aircraft communications addressing and reporting system (ACARS) protocols [48]. Their technique was evaluated on 5,157 and 3,143 classes and sample sizes of 13,000,000 and 900,000 transmissions, respectively, and achieves 96.3% accuracy for ADS-B and 98.1% accuracy for ACARS, demonstrating both the degree of variability between transmitters and the ability of a neural network model to distinguish

individual transmitters in such a large population.

Wong demonstrated a series of CNN-based techniques for the RF fingerprinting problem [49]. The first technique uses a CNN model as a feature extractor to measure I/Q imbalance parameters of a recorded transmitter, then uses these parameters as input to a Bayesian model to perform classification based on a learned decision boundary. The second technique uses a CNN model as a feature extractor, then uses the density-based spatial clustering of applications with noise (DBSCAN) algorithm to perform classification. Our approach differs in several important ways, including that our approach uses a neural network for both feature extraction and classification. Additionally, our approach synchronizes all transmissions prior to presentation to the neural network to prevent an intermediate adversary from fooling the classifier by performing a frequency shift with their own device, as we demonstrate in Chapter 5. Finally, their experiments demonstrate the technique using identical transmissions from all radios, while the work presented here assumes unique random bit sequences are encoded in all transmissions.

Hanna and Cabric developed a technique to simulate realistic nonlinear power amplifiers from experimental measurements of real devices and demonstrated both CNN and MLP networks to identify which simulated transmitter within a population of 500 produced a given transmission [50]. Furthermore, they evaluated the impact that the input representation of discrete Fourier transform (DFT) coefficients has on discriminability and determined that the magnitude-only representation achieves higher performance than either cartesian or polar coordinates. Chatterjee *et al.* similarly evaluated performance using simulated transmitters, and estimated that they

can reliably differentiate 10,000 unique devices using a hand-crafted feature vector of device-specific features and channel information using a MLP for classification [51]. However, the performance of both techniques on real devices remains an open question.

Restuccia *et al.* analyzed the problem of reusing a trained RF fingerprint identification network when the channel changes between the transmitter and receiver [52]. To address a demonstrated performance drop-off after changes to the channel, they adaptively learn a finite impulse response (FIR) filter at the receiver which is fed back to the transmitter and used to adjust their signal prior to transmission. They demonstrated that in doing so, they can increase identification accuracy while also reducing the effectiveness of an adversary which steals the filter coefficients. Morin *et al.* attempted to achieve channel-invariance through diversity of their training data by recording from fixed devices with a robot moving randomly throughout the room to continuously change the channel [53]. They demonstrated improved generalization to a new channel, achieved by placing a metallic stool in the room, compared to training in a completely static environment. We explore a similar option in Chapter 5 where we simulate multipath environments and train on a diverse set of individual channels to attempt to achieve invariance.

2.3 Other Applications of Neural Networks in the RF Domain

Benvenuto *et al.*, as well as Mkadem and Boumaiza, demonstrated neural network methods for digital predistortion of transmitted signals to compensate for

RF power amplifier nonlinearity [54,55]. Additionally, Wray and Green proved that a certain class of neural network models can equivalently represent a Volterra kernel which is commonly used to model the nonlinear memory effects that are present in RF power amplifiers [56]. Rather than compensating the distortions present in the transmitted signals, RF fingerprinting algorithms seek to use them for classification.

Weng *et al.* used a simple MLP network to identify the presence or absence of a receiver based on its unintended emissions [57]. They found that the oscillator on-board each receiver tended to couple to the antenna, causing a weak signal to be radiated. To detect this signal, they created a feature vector from cross-correlation values between the measured signal and a reference signal in multiple frequency bands, and used the MLP to classify the device presence or absence.

Wang *et al.* presented several successful techniques for performing indoor localization using channel state information (CSI) from multiple antennas and subcarriers in an orthogonal frequency-division multiplexing (OFDM) network to create feature vectors, and utilizing several neural network structures including autoencoders, restricted Boltzmann machines, and ResNets for feature extraction and location estimation [58–60]. Zhang *et al.* similarly demonstrated a wireless localization technique which uses a stacked denoising autoencoder [61] to pretrain a deep neural network for coarse localization, and a hidden Markov model (HMM) for fine-grained localization [62]. In a similar manner, Huang *et al.* showed that a deep neural network can be used to perform direction of arrival estimation as well as channel estimation in massive multiple-input multiple-output (MIMO) systems [63].

Mendis *et al.* developed a deep belief network-based modulation recognition

algorithm operating on images produced using the spectral correlation function, a measurement derived directly from the recorded signal [64]. Similarly, O'Shea *et al.* demonstrated success in applying CNNs to the problem of blind modulation classification [65]. Their method, which is based on ResNets, operates directly on fixed-length complex baseband signals and learns to reliably identify the modulation scheme used in the input signal in the presence of realistic channel effects such as carrier phase/frequency offsets, Rayleigh channels, and noise. Later work has shown that an adaptation of spatial transformer networks [66] can be used to perform blind time, phase, and frequency synchronization on the signals in advance, increasing the accuracy of the modulation recognition system [67].

Schmidt *et al.* used a CNN model for wireless interference identification for IEEE 802.11, 802.15.4, and 802.15.1 devices in order to classify wireless devices by wireless standard and allocated frequency channel [68]. Bitar *et al.* similarly demonstrated the ability of a CNN model to identify transmissions by standard using time-frequency measurements over the same set of protocols, including the ability to correctly classify when devices of multiple standards were transmitting simultaneously [69]. Kokalj-Filipovic *et al.* showed that pretraining using stacked denoising autoencoders can reduce the size and increase the accuracy of a fully connected network for protocol identification over a superset of the same protocols [70].

Akeret *et al.* developed a CNN approach to radio frequency interference mitigation using a U-Net [71] architecture to identify and segment sources of man-made radio interference from astronomical radio signals in time-frequency input data [72].

Rutagemwa *et al.* demonstrated a generative model based on RNNs to generate signals with similar spectral usage statistics to a real environment, and used this model to enhance spectrum assignment and sharing approaches for land mobile radios [73].

Recently, a handful of results for adversarial machine learning techniques in the RF domain have been published. Sadeghi *et al.* developed both white box and black box attacks on a deep learning-based modulation classification algorithm and showed that they can successfully force an incorrect classifier decision with high probability, even, in the case of the white box attack, when the perturbation is several orders of magnitude lower in power than the noise [74]. Kokalj-Filipovic *et al.* have published a number of works in this domain [75–77]. They demonstrated a successful attack on both modulation and protocol recognition neural networks using a surrogate model of the classifier on the attack side. Furthermore, they show that pretraining convolutional weights in a CNN classifier with an autoencoder can effectively defend against this sort of attack by essentially filtering out adversarial perturbations since they are not useful for reconstruction. Finally, they show that Kolmogorow-Smirnov tests on either the distribution of peak-to-average power ratio measurements or softmax output layer probability distributions can detect statistical differences between real and adversarial examples for these sorts of attacks against classifiers in the RF domain.

GAN techniques in particular have found several recent uses within the RF domain. Davaslioglu and Sagduyu developed a conditional GAN technique to develop additional training data for random forest and SVM classifiers for spectrum sensing [78]. Additionally, they developed a technique based on bidirectional GANs

for performing domain adaptation when the channel environment changes. Tang *et al.* used GANs to increase the quality of training data for a modulation classification algorithm in order to increase the robustness of their classifier [79]. O'Shea *et al.* developed a GAN model to approximate the response of a communication channel and used this model to find an optimal physical layer encoding scheme for that learned channel [80]. Likewise, Ye *et al.* developed a GAN model to represent channel impairments, then used this model to help train an end-to-end communication system where the transmitter may utilize CSI to optimize their transmissions [81].

Chapter 3: Background

This chapter provides a brief introduction to devices that support the IEEE 802.15.4 standard, such as ZigBee devices, and also outlines the likely sources of imperfection, and thus uniqueness, in the front-end of an example transmitter.

3.1 IEEE 802.15.4 and ZigBee

The IEEE 802.15.4 specification [1] defines the physical and MAC standards used by a number of wireless protocols, including ZigBee. While some of the devices used in our experiments implement the full ZigBee PRO stack, ZigBee is a higher level protocol built on top of the physical layer specifications defined in the IEEE 802.15.4 standard [1, 2]. The standard specifies a number of different physical layer schemes; the scheme which we target specifies offset QPSK (O-QPSK) with half-sine pulse shaping and direct-sequence spread spectrum (DSSS) modulation at a data rate of 250 Kbps.

The half-sine pulse shaping used allows the data to be demodulated noncoherently using a binary frequency shift keying (FSK) demodulator. Because of the DSSS technique used, 32 symbols are transmitted for every data byte. The first five bytes of any IEEE 802.15.4 O-QPSK transmission include a four byte preamble of

zeros, followed by the one byte start frame delimiter (SFD) 0xA7. These five bytes collectively form the SHR field of the packet.

IEEE 802.15.4 compliant devices may operate in a number of different frequency bands depending on the region, including 868, 915, and 2450 MHz bands. The devices studied in this dissertation operate exclusively in the 2450 MHz industrial, scientific, and medical (ISM) band and support 16 channels centered within 2405 MHz to 2480 MHz, each separated by 5 MHz. These devices must output signals with an error vector magnitude (EVM) of less than 35% and must support a transmit power level of at least -3 dBm.

3.2 Sources of Transmitter Variability

Variability in a number of different components causes measurable effects that make it possible to perform RF fingerprinting. This section will outline the different effects that may be present in transmitted signals and associate each effect with the RF front-end components that cause them, using the direct-conversion transmitter architecture shown in Figure 3.1 as an example. While a number of different transmitter architectures are possible, no architecture is able to avoid all of the described effects entirely.

Note that due to the tolerances in physical layer standards, such as the 35% maximum EVM permitted in IEEE 802.15.4 devices, these effects can be present in transmitted signals which still demodulate correctly and meet the spectral requirements. As such, some of the variability between devices is a result of intentional

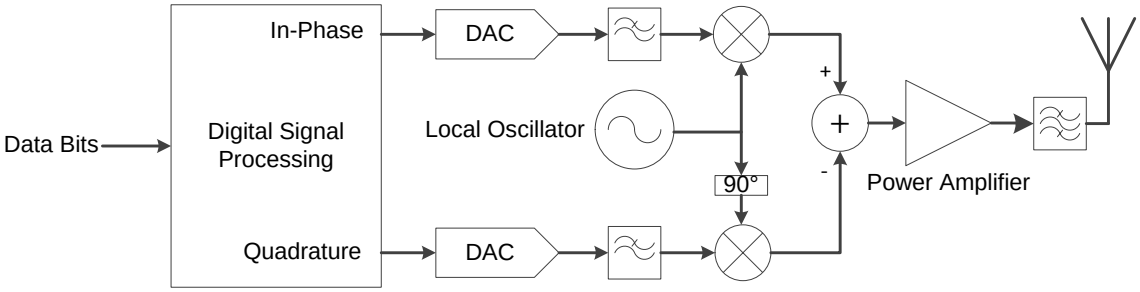


Figure 3.1: A diagram showing the components in the front-end of a direct-conversion transmitter.

decisions made by designers of different device models, such as the specific low-pass filters to apply to the I and Q channels. However, variability can also be seen in devices of the same model and is a result of unintentional differences that occur during the manufacturing process.

3.2.1 Power Amplifier Nonlinearity

An ideal power amplifier has a constant, frequency-independent gain such that the output signal is simply the input signal at a higher power level. This behavior is deemed *linear* because a plot of the input power level versus output power level is simply a straight line where the slope corresponds to the linear gain. In practice, power amplifiers are nonlinear devices that behave approximately linearly below some threshold of input power, and nonlinear beyond that threshold as the amplifier approaches saturation. An example of this behavior is shown in Figure 3.2 and demonstrates the input/output relation for both an ideal linear amplifier and a nonlinear amplifier.

The effect of a nonlinear power amplifier on a signal may be illustrated by passing a sum of tones through the device [82]. The nonlinearity in the amplifier

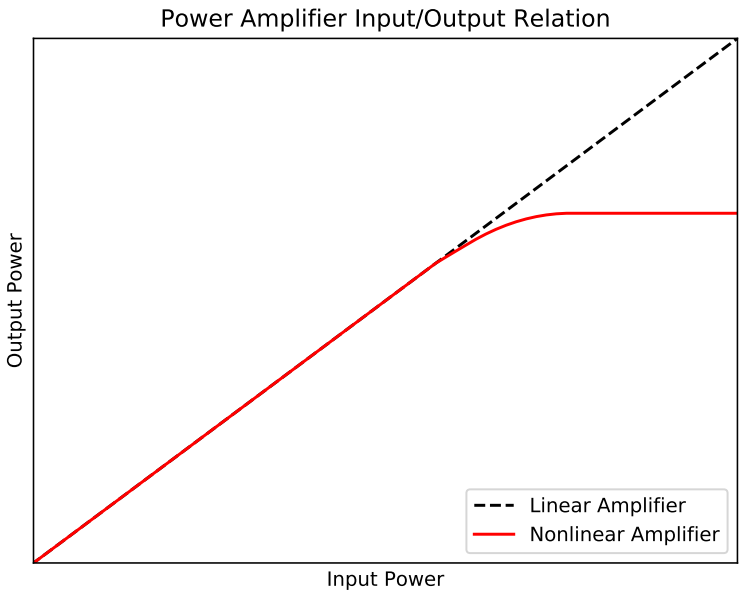
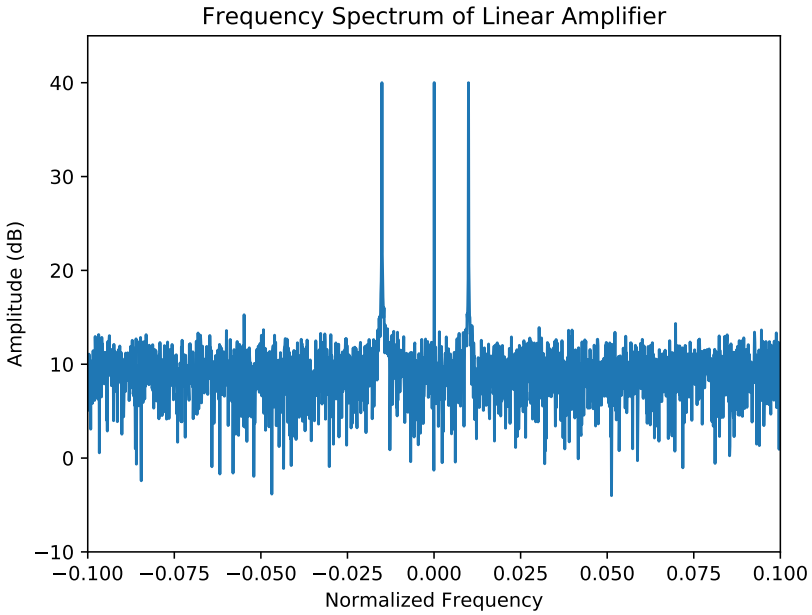


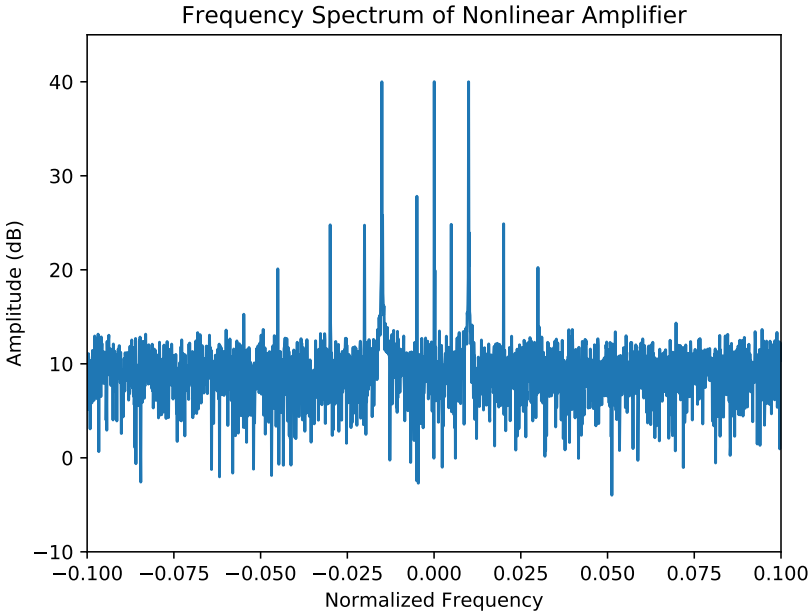
Figure 3.2: An example input/output relation for a nonlinear power amplifier. At lower input power levels, the output power is a scalar multiple of the input power. Once the input power passes a certain threshold, the amplifier begins to operate in the nonlinear region and eventually saturates.

causes an effect known as *intermodulation* which induces additional tones in the output spectrum that were not present in the input. These tones are at predictable locations given by sums of integer multiples of the input frequencies. An example of this effect is shown in Figure 3.3 where Figure 3.3(a) represents a linear amplifier and simply shows the three tones that were present at the input, and Figure 3.3(b) represents a nonlinear amplifier where a number of additional tones are present. *Harmonic distortion* may be visualized by passing a single tone through the amplifier, causing additional tones at integer multiples of the input frequency.

Constant envelope modulation schemes, such as the O-QPSK with half-sine pulse shaping modulation employed by IEEE 802.15.4, are more robust to nonlinear amplifiers because they operate at a fixed point on the input/output curve [83]. However, other sources of distortion in the RF front-end, such as I/Q imbalance,



(a) Linear Amplifier Frequency Spectrum



(b) Nonlinear Amplifier Frequency Spectrum

Figure 3.3: The frequency spectrum for a set of three tones through a simulated (a) linear amplifier and (b) nonlinear amplifier. The nonlinear amplifier produces intermodulation as evidenced by the additional tones present in its spectrum.

may cause the envelope to be non-constant prior to the power amplifier. Since communication signals are typically present over a much larger number of frequencies than the sum of three tones, the effect on the spectrum is less straightforward than additional induced tones.

Power amplifier nonlinearity is often modeled using a subset of the causal complex baseband Volterra series [84, 85], given by

$$y(t) = \sum_{k=0}^{\infty} \int_0^{\infty} \dots \int_0^{\infty} h_{2k+1}(\tau_1, \dots, \tau_{2k+1}) \prod_{m=1}^{k+1} x(t-\tau_m) \prod_{n=k+2}^{2k+1} x^*(t-\tau_n) d\tau_1 \dots d\tau_{2k+1}, \tag{3.1}$$

where $y(t)$ is the model output, $x(t)$ is the complex baseband signal, and coefficients $h_{2k+1}(\tau_1, \dots, \tau_{2k+1})$ characterize linear and nonlinear behavior of the particular system. In practice, this series must be truncated, requiring the order, $2k+1$, to be limited to a finite maximum value and delay terms τ_i to be limited, $\forall i$, so that the system has finite memory. Since even finite-order, finite-memory Volterra models are computationally expensive, subsets such as memory polynomials [86] and generalized memory polynomials [87] may be used.

3.2.2 I/Q Imbalance

I/Q imbalance comprises two issues: *gain imbalance* and *quadrature skew*. Gain imbalance occurs as a result of mismatch in the components in the I and Q paths, whether in the digital-to-analog converters (DACs), low-pass filters, or mixers, and causes unequal gain along these two paths. Quadrature skew occurs when the carrier signals input to the mixers are not exactly 90° out of phase [88].

The combined effects of these two issues may be seen in Figure 3.4, where a 16-symbol quadrature amplitude modulation (QAM) constellation is used to highlight the impact. Gain imbalance causes the constellation to be stretched or squeezed along the I and Q dimensions, and quadrature skew causes a loss of orthogonality between I and Q channels.

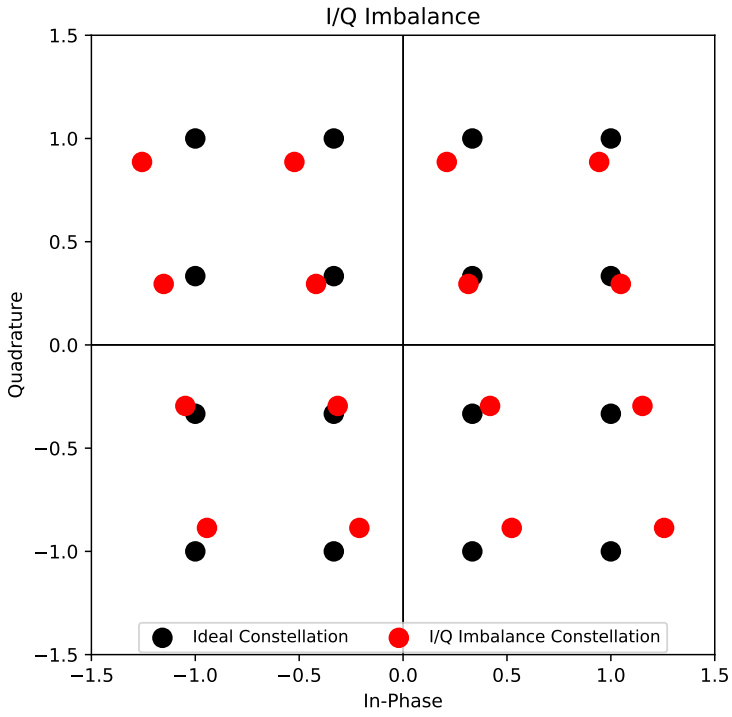


Figure 3.4: The effect of I/Q imbalance on the constellation diagram of a 16QAM modulation scheme.

Define $x(t) = i(t) + jq(t)$ as the ideal baseband signal at time t. Then, the ideal output of the adder after mixing is given by

$$y(t) = i(t)\cos(2\pi ft) - q(t)\sin(2\pi ft), \tag{3.2}$$

where f is the center frequency at which the signal is transmitted.

Let α denote excess gain along the I path, β denote excess gain along the Q path, and ϕ denote quadrature skew which we model as phase error in the Q path. Then, the output of the adder is

$$y(t) = (1+\alpha)i(t)\cos(2\pi ft) - (1+\beta)q(t)\sin(2\pi ft + \phi). \tag{3.3}$$

Using the following trigonometric identity,

$$\sin(\theta_1 + \theta_2) = \sin(\theta_1)\cos(\theta_2) + \cos(\theta_1)\sin(\theta_2), \tag{3.4}$$

this becomes

$$y(t) = (1+\alpha)i(t)\cos(2\pi ft) - (1+\beta)q(t)\Big[\sin(2\pi ft)\cos(\phi) + \cos(2\pi ft)\sin(\phi)\Big], \tag{3.5}$$

which can be rewritten as

$$y(t) = \Big[(1+\alpha)i(t) - (1+\beta)q(t)\sin(\phi)\Big]\cos(2\pi ft) - \Big[(1+\beta)q(t)\cos(\phi)\Big]\sin(2\pi ft), \tag{3.6}$$

where the loss of orthogonality between I and Q channels is now apparent. From inspection, it can then be seen that this is equivalent to a transmitter where the

baseband signal is given by $x_{\text{imb}}(t) = i_{\text{imb}}(t) + jq_{\text{imb}}(t)$, where

$$\begin{aligned} i_{\text{imb}}(t) &= (1+\alpha)i(t) - (1+\beta)q(t)\sin(\phi), \\ q_{\text{imb}}(t) &= (1+\beta)q(t)\cos(\phi). \end{aligned} \tag{3.7}$$

3.2.3 Phase and Frequency Error

In practice, the local oscillators (LOs) used for mixing at the transmitter and receiver are not perfectly synchronized, resulting in phase and frequency offsets between the two devices. In addition, oscillators exhibit a time-varying additive random noise to their phase known as *phase noise*. As a result, the LO for the transmit side produces a signal given by

$$\cos(2\pi(f+\delta_T)t + \phi_T + \theta_T(t)), \tag{3.8}$$

and the LO for the receiver produces a signal given by

$$\cos(2\pi(f+\delta_R)t + \phi_R + \theta_R(t)), \tag{3.9}$$

where δ_T and δ_R represent frequency offsets, ϕ_T and ϕ_R represent fixed phase offsets, and $\theta_T(t)$ and $\theta_R(t)$ represent time-dependent phase noise for the transmitter and receiver, respectively.

3.2.4 Linear Filters

The architecture given in Figure 3.1 includes linear filters: low-pass filters on both the I and Q paths, and a band-pass filter immediately prior to transmission. As all three of these filters are implemented in the analog domain, they are all susceptible to manufacturing differences due to non-zero tolerance in the components they are made from, such as resistors, capacitors, inductors, and transistors. Moreover, the specific filter design is often a decision that is left to the manufacturer, provided that specific power spectral density requirements are met. These filters may each be modeled as a causal linear time-invariant (LTI) system for which the output is simply the convolution of the input signal and the impulse response of the filter, given by

$$y(t) = \int_0^\infty h(\tau)x(t-\tau)d\tau, \tag{3.10}$$

where $y(t)$ is the filter output, $x(t)$ is the filter input, and $h(t)$ is the impulse response of the filter.

3.2.5 I/Q Origin Offset

If there if any direct current (DC) offset present at the output of the DACs, this offset will be presented as a tone at the center frequency after mixing. After downconversion at the receiver, this tone will be shifted to 0 Hz and cause the center of the constellation diagram to shift from the origin, resulting in symbols that are likewise shifted from their ideal locations, as shown in Figure 3.5. This may also

be caused by mismatch between the mixers in the I and Q paths resulting in LO leakage to the transmitted signal [88]. The I/Q origin offset may be modeled as simple shifts in $i(t)$ and $q(t)$ from their ideal values, or

$$y(t) = (i(t) + \Delta_i) + j(q(t) + \Delta_q), \tag{3.11}$$

where Δ_i and Δ_q represent the offset in the I and Q directions, respectively [89].

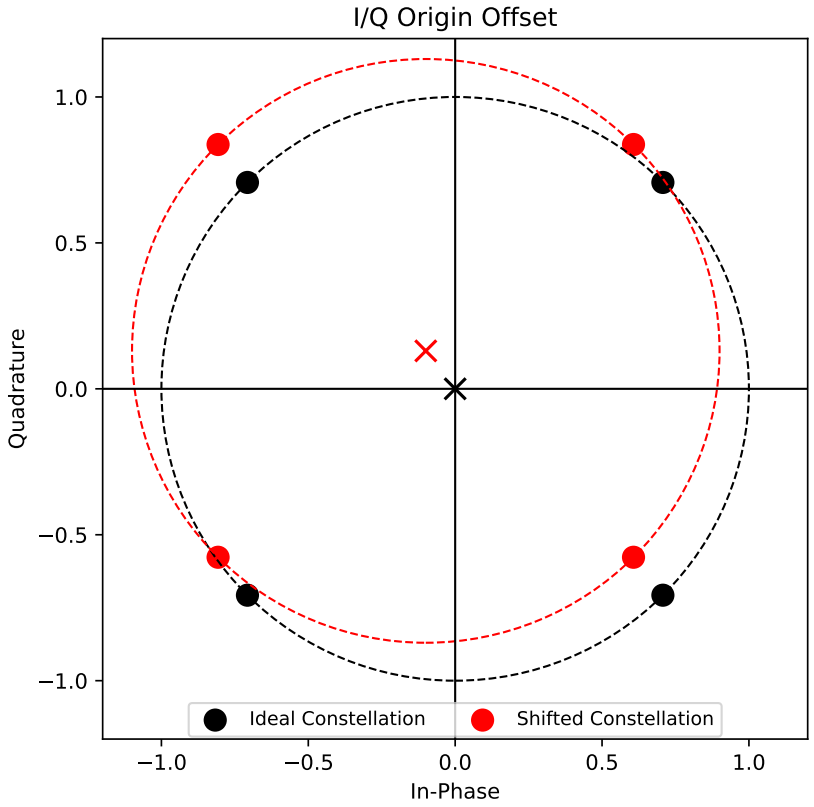


Figure 3.5: The effect of I/Q origin offset on the constellation diagram of the transmitted QPSK symbols. The symbols, which lie on a circle to denote constant amplitude, are shifted in both the I and Q directions.

3.2.6 Digital-to-Analog Converter Nonlinearity

DACs map a finite number of digital codes to corresponding output voltages. An N-bit signed linear DAC may have output defined according to

$$V_{\text{out}}(d_{\text{input}}) = G\frac{d_{\text{input}}}{2^N}, \tag{3.12}$$

where V_{out} represents the output voltage, d_{input} represents the signed decimal number associated with the input code, and G is a constant scale factor. The output of a linear DAC, therefore, changes by a fixed voltage when the input changes to an adjacent code. However, a nonlinear DAC does not obey this simple relationship and can be evaluated by its *differential nonlinearity* (DNL) and *integral nonlinearity* (INL).

Differential nonlinearity measures the deviation of the actual change in voltage between adjacent input codes from the ideal value, and will vary depending on which input codes are evaluated for a nonlinear DAC. Integral nonlinearity measures the deviation of analog output levels from the ideal level, and also is dependent on which input code is evaluated [90]. An example of input/output relations for both linear and nonlinear DACs is shown in Figure 3.6. Similarly to power amplifier nonlinearity, DAC nonlinearity introduces harmonic distortion and intermodulation into the output signal [91], and may likewise be modeled using Volterra series representations [92].

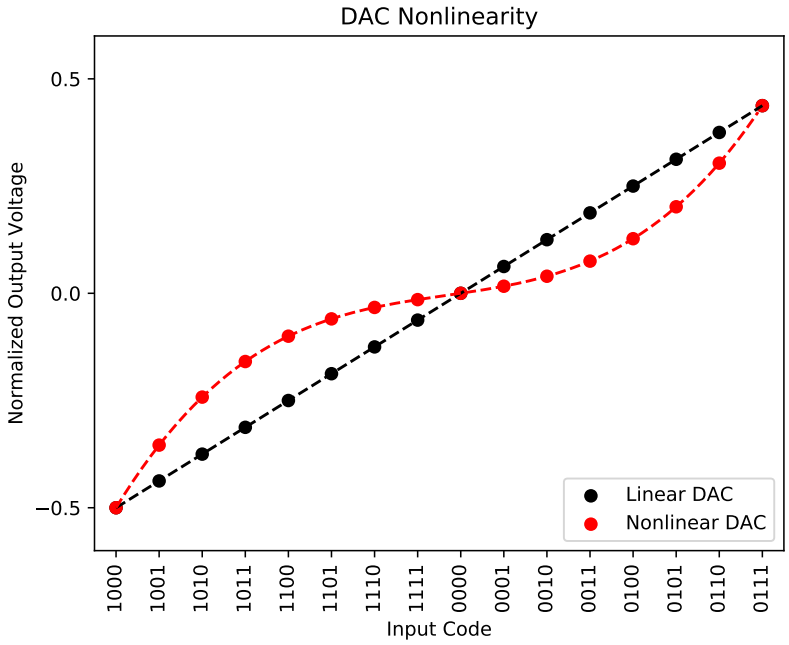


Figure 3.6: Example input/output relations for linear and nonlinear 4-bit signed DACs. Note that the input codes are represented in two's complement form.

Chapter 4: Convolutional Neural Networks for Identification and Verification of RF Fingerprints from Error Signals

4.1 Introduction

First, we present an RF fingerprinting technique which employs a CNN architecture to classify windowed sections of recorded complex baseband signals by transmitting device. We develop a careful pre-processing sequence to prepare recorded signals for presentation to the network and focus on removing sources of potential bias or trivial classification to improve classifier robustness. We then present our CNN architectures, which are structured similarly to networks typically used in the computer vision domain. Then, we show that combining network predictions across multiple windows enables classification of each full transmission, and further demonstrate that the classification performance on each full transmission is significantly higher than the single window performance. This section presents classifiers for both the verification and identification problems, and demonstrates the performance on data recorded both in a laboratory environment and an urban outdoor environment from a set of seven commercial ZigBee transmitters.

4.2 Methods

4.2.1 Data Collection

We demonstrate our technique on a collection of seven Digi XBP24CZ7SITB003 ZigBee Pro devices operating in the 2.4 GHz band. The device under test (DUT) was added to a network with another device, known as the coordinator in the ZigBee protocol, to allow transmission. A unique set of packets with random 32 byte payloads was generated in MATLAB for each device and sent to the DUT using the Digi XCTU software [93]. All devices transmitted on IEEE 802.15.4 channel 11, corresponding to a 2.405 GHz center frequency, and were configured to transmit at their maximum power level of 18 dBm. For each device, approximately 28 seconds of data was digitized at a sampling rate of 16 MHz, and the received signal was downconverted to baseband and recorded on a Rohde & Schwarz FSW67 signal and spectrum analyzer with external SRS FS725 reference.

We demonstrate our technique via two experiments: In one experiment, a CNN is trained on data collected outdoors through a noisy channel. In a second experiment, a CNN is trained on data collected in a lab environment with simulated additive white Gaussian noise. For the outdoor experiment, both the coordinator and the DUT were connected to Pulse Electronics W1030 omni-directional antennas and DUTs were located approximately 30 meters from the receive antenna in an urban outdoor environment where the estimated in-band SNR of the received transmissions is approximately 28 dB. For the lab experiment, DUTs were hardwired to

a power splitter which was connected directly to the signal and spectrum analyzer, and the transmissions are assumed to be virtually noise-free.

4.2.2 Data Pre-processing

Prior to training the network, a number of pre-processing steps are taken in MATLAB. First, each individual transmission is extracted from the recording. Next, the estimated ideal signal is used to synchronize the transmissions in phase, frequency, and time. Because the signals are oversampled, each signal is then low-pass filtered to remove components that fall outside the transmitter passband. Finally, the error signal for each transmission is calculated and saved to its own file. The error signal for each transmission is used as an input to the CNN.

4.2.2.1 Extraction

Approximate start and stop times for each transmission were detected in MATLAB by thresholding the amplitude differences between adjacent samples. Each transmission was aligned to a reference signal containing the five byte SHR by cross-correlating the instantaneous frequency of the received and reference signals. The final alignment selected the time lag between signals that maximized the cross-correlation and produced a valid checksum of the demodulated data. Transmissions from the coordinator were discarded, and to demonstrate that the method is unaffected by the data contents of the transmission, only the portion of the signal containing the 32 byte random payload and the two byte checksum from each trans-

mission was retained. Each remaining signal was scaled by dividing all samples by the root mean square (RMS) of the magnitude signal so that slight power level differences between transmitters did not permit trivial classification. Each of the first 1,000 valid transmissions from each device for each experiment was processed in this manner and saved to its own MATLAB file.

4.2.2.2 Additive Noise

For the lab data experiment, each transmission was included in the dataset several times with varying levels of simulated additive white Gaussian noise (AWGN). After extraction, additive noise was imposed on each transmission via MATLAB's *awgn()* function. Each transmission was included in the dataset eight times for each SNR in the range $\{10, 15, 20, \ldots, 40\}$ dB. Each instance of the same transmission at a particular SNR uses a different random noise signal. By including each transmission several times, we are able to increase the amount of training data given a fixed amount of lab data. The network was trained on the entire training dataset at once rather than training at a single SNR.

4.2.2.3 Synchronization

A common problem in demodulation of wireless transmissions is that of synchronization between the transmitter and receiver. In this case, the signals may be demodulated noncoherently. However, because we are using the baseband error signal, we perform synchronization on each transmission individually to remove

fixed frequency and phase offsets from the signal and improve the sample timing between the transmitter and receiver. In doing so, we aim to remove trivial features that fluctuate randomly or are easily spoofed, and learn more reliable features for classification. Further, this reduces sources of bias in the experiment since the phase relationship between the transmitter and receiver oscillators could change by simply resetting either device.

Since the transmissions can be demodulated noncoherently and we are targeting a known modulation scheme, we assume that the ideal discrete-time baseband signal can be estimated based on the demodulated symbols. We model the synchronization error between the transmitter and receiver as a fixed frequency and phase offset which are constant for the entirety of a single transmission, but which may be different for each transmission. For simplicity, we model these as offsets in the transmitter from the receiver's ideal values. That is, for complex baseband signal $x_{\text{ideal}}(t)$, time index t, and assuming no other channel distortion, the ideal passband signal is given by

$$y_{\text{ideal}}(t) = \text{Real}(x_{\text{ideal}}(t)e^{j\omega_c t}), \tag{4.1}$$

where ω_c is the ideal carrier frequency. Due to the presence of the phase and frequency offset, the actual transmitted signal is given by

$$y_{\text{sent}}(t) = \text{Real}(x_{\text{ideal}}(t)e^{j((\omega_c+\omega_o)t+\theta_o)}), \tag{4.2}$$

where ω_o and θ_o are the transmitter's frequency and phase offset, respectively. After downconversion to baseband at the receiver, the received signal is

$$x_{\text{meas}}(t) = x_{\text{ideal}}(t)e^{j(\omega_o t+\theta_o)}. \tag{4.3}$$

To perform carrier synchronization, our goal is to find frequency correction ω_f and phase correction θ_f such that

$$x_{\text{meas}}(t)e^{j(\omega_f t+\theta_f)} \approx x_{\text{ideal}}(t). \tag{4.4}$$

If we model $x_{\text{ideal}}(t)$ as an arbitrary time-dependent signal in the complex domain, such that $x_{\text{ideal}}(t) = A(t)e^{j\theta(t)}$, then this becomes

$$A(t)e^{j(\theta(t)+\omega_o t+\theta_o+\omega_f t+\theta_f)} \approx A(t)e^{j(\theta(t))}. \tag{4.5}$$

If we define $\angle x_{\text{ideal}}(t)$ and $\angle x_{\text{meas}}(t)$ to be the phase angle of the ideal and measured baseband signals, respectively, then the following two expressions are equivalent

$$-\omega_f t - \theta_f \approx (\theta(t) + \omega_o t + \theta_o) - \theta(t), \tag{4.6}$$

$$\omega_f t + \theta_f \approx \text{unwrap}(\angle x_{\text{ideal}}(t) - \angle x_{\text{meas}}(t)). \tag{4.7}$$

We use linear regression to find ω_f and θ_f that minimize the following expression

$$\underset{\omega_f,\theta_f}{\text{argmin}} \sum_t \left(\omega_f t + \theta_f - \text{unwrap}(\angle x_{\text{ideal}}(t) - \angle x_{\text{meas}}(t))\right)^2. \tag{4.8}$$

Synchronization is typically performed with phase-locked loops to recover the carrier prior to demodulation. In this case, since the signal may be demodulated noncoherently and the estimated ideal signal may be generated, linear regression is used to find optimal values of ω_f and θ_f and to perform the frequency and phase corrections for each transmission.

Finally, the sample timing of each transmission is synchronized to the estimated ideal signal. To start, the estimated ideal signal is generated at a sample rate of 160 MHz and the received signal is interpolated by a factor of ten. The two signals are aligned by maximizing the cross-correlation over possible lag values in the range $[-9, 9]$. Rather than shifting the estimated ideal signal, the received signal is always shifted and, if necessary for alignment, the first ten samples of the estimated ideal signal are removed. Both signals are then downsampled by a factor of ten to restore the original 16 MHz sample frequency. By only shifting the received signal, the sample timing of the estimated ideal signal is unchanged and at most one sample is lost.

4.2.2.4 Error Signal Generation

Each signal was forward-backward filtered through a fourth order Butterworth filter with a 2 MHz passband. The ideal signal was subtracted from each transmission resulting in the error signal for that transmission.

Any device-dependent features that may be useful for classification will not be present in the ideal signal. By subtracting the estimated ideal signal from the

recorded transmissions to generate the error signal, we allow the network to ignore the portion of the signal that would be common across devices and focus on the differences which may be caused by features specific to each device, such as those outlined in Chapter 3.

4.2.2.5 Dataset Generation

Each signal was split into N non-overlapping segments of M consecutive samples. The network was trained on and learned to classify these segments of M samples, and the output of the network on each segment was combined later to classify each overall transmission. This splitting technique is useful for multiple reasons. For one, reducing the length of the network input signal decreases the size of the network, allowing for faster training and likely requiring less training data than a larger network. In addition, by splitting the signal into many smaller components, the method can be adapted for use with signals of arbitrary lengths by simply adjusting the threshold for classification by the number of M-sample segments in the signal.

The complex data from the time-series error signal was split into real and imaginary parts and treated as two separate input channels. Each time-series signal was normalized by subtracting off the mean value and dividing by the standard deviation, computed separately on both the real and imaginary parts for a given transmission. One dataset was created for the outdoor experiment, and a second for the indoor experiment. Each full dataset of 7,000 transmissions was randomly

Layer	Dimension	Parameters	Activation
Input	1,024x2	-	-
Convolution 1D	128x19	4,992	ELU
Max Pooling	2	-	-
Convolution 1D	32x15	61,472	ELU
Max Pooling	2	-	-
Convolution 1D	16x11	5,648	ELU
Max Pooling	2	-	-
Flatten	-	-	-
Dense	128	239,744	ELU
Dropout (0.5)	-	-	-
Dense	16	2,064	ELU
Dropout (0.5)	-	-	-
Dense	7	119	Softmax

Table 4.1: The layers of the identification network for the outdoor data, along with the number of parameters and activation function of each layer.

partitioned into 80% training data, 10% validation data, and 10% testing data with all N segments of a given transmission belonging to the same set. The same set assignment was used for both the identification and verification problems for a given dataset. The order of the segments was randomly shuffled for the training dataset, and then the resulting datasets were saved in HDF5 format.

4.2.3 Convolutional Neural Network

A deep CNN was used to solve the identification and verification problems, with the network structures given in Tables 4.1, 4.2, and 4.3. The exponential linear unit (ELU) [94] was selected as the activation function for all applicable layers except the output layer where Softmax was used. The identification networks for the outdoor experiment and lab experiment have a total of 314,039 and 1,074,727 trainable parameters, respectively. The verification networks for both experiments

Layer	Dimension	Parameters	Activation
Input	1,024x2	-	-
Convolution 1D	128x19	4,992	ELU
Max Pooling	2	-	-
Convolution 1D	32x15	61,472	ELU
Max Pooling	2	-	-
Flatten	-	-	-
Dense	128	999,552	ELU
Dropout (0.5)	-	-	-
Dense	64	8,256	ELU
Dropout (0.5)	-	-	-
Dense	7	455	Softmax

Table 4.2: The layers of the identification network for the lab data with artificial noise, along with the number of parameters and activation function of each layer.

Layer	Dimension	Parameters	Activation
Input	1,024x2	-	-
Convolution 1D	32x19	1,248	ELU
Max Pooling	2	-	-
Convolution 1D	128x19	77,952	ELU
Max Pooling	2	-	-
Convolution 1D	32x15	61,472	ELU
Max Pooling	2	-	-
Convolution 1D	16x11	5,648	ELU
Max Pooling	2	-	-
Flatten	-	-	-
Dense	128	106,624	ELU
Dropout (0.5)	-	-	-
Dense	16	2,064	ELU
Dropout (0.5)	-	-	-
Dense	2	34	Softmax

Table 4.3: The layers of the verification networks, along with the number of parameters and activation function of each layer.

have a total of 255,042 trainable parameters. Categorical cross-entropy was used to compute the loss.

For these experiments, there are $K = 7$ devices, $N = 17$ segments per transmission, and $M = 1,024$ samples per segment. Values for N and M were determined experimentally. Intuitively, $M = 1,024$ corresponds to segments of length two bytes. As demonstrated in our results, this choice of segment length is long enough to detect repetitive patterns in the transmissions. However, this choice of M is also short enough to permit a relatively small input layer, and thus a CNN of a manageable size. Finally, M corresponding to a two byte segment allows for the scalability of the method to any signal with a length that is a multiple of two bytes. Therefore, at most one byte from an arbitrary transmission would be discarded for classification purposes.

For identification, a network was trained on all seven devices, with each device representing a class. For verification, the network was trained on all seven devices, with one device selected as the positive class for each model and the remaining devices all considered as a single negative class.

4.2.3.1 Classification

For the identification problem with K devices, the network output is a length-K vector interpreted as the set of estimated probabilities that the selected M-sample error signal belongs to each of the K devices. For verification, the network output is a vector of length two, representing the estimated probability distribution of the

event that the signal matches the device associated with the positive class for that network.

To assign a class label to the overall transmission x, we treat each of the M-sample vectors x_i for $i = 1, \ldots, N$ as mutually independent. Of course, this assumption is overly simplistic, as all M-sample vectors belong to the same transmission. Still, it provides a simple way to combine the estimated probability distributions of the different segments into a single metric and also provides strong classification performance. With this assumption, and denoting the estimated probability that segment x_i belongs to class ω_k as $p(\omega_k; x_i)$, the estimated probability that all segments x_i, $i = 1, \ldots, N$ belong to class ω_k is given by

$$p(\omega_k; x_1, \ldots, x_N) = \prod_{i=1}^{N} p(\omega_k; x_i). \tag{4.9}$$

Since it is known a priori that all x_i for $i = 1, \ldots, N$ come from the same transmission x, and thus belong to the same class, the estimated probability $p(\omega_k; x)$ that transmission x belongs to class ω_k is given by

$$p(\omega_k; x) = \frac{p(\omega_k; x_1, \ldots, x_N)}{\sum_{r=1}^{K} p(\omega_r; x_1, \ldots, x_N)}. \tag{4.10}$$

Transmission x may then be assigned to the class with the highest estimated probability. Because of numerical precision concerns when a large number of segments

are used, we equivalently assign x to the class ω_k which maximizes:

$$\sum_{i=1}^{N} \log(p(\omega_k; x_i)). \tag{4.11}$$

4.2.3.2 Training

For the identification problem, the proportion of samples in each class was approximately 1/7, with some variation due to the randomness of the split. Since verification is a one-vs-all problem, approximately 1/7 of the available data belonged to the positive class, and 6/7 of the data belonged to the negative class for each verification problem. During identification, all classes were given uniform weight. During verification, the positive class was given a weight of six and the negative class was given a weight of one due to the class imbalance.

Glorot uniform initialization [95] was used for kernel initialization of all convolutional and dense layers, and all bias vectors were initialized to the zero vector. The Adam optimizer and its original parameters was used for training [96]. The batch size was set to 1,024 segments and the order of the batches was shuffled prior to each epoch.

During training, only the classification performance of each 1,024-sample segment is considered. Each overall transmission is not classified during the training phase as the loss is computed for the segments only.

Validation was performed after every epoch. For the identification problem, a classification of each transmission in the validation set is performed during valida-

tion using Equation 4.11 as the confidence metric for each class. The classification accuracy on the validation set was tracked, and training was terminated when the accuracy did not improve for R consecutive evaluations on the validation set, at which point the best performing model was saved and used for testing. For the outdoor identification problem $R = 20$, and for the outdoor verification problem $R = 10$. For the identification problem with lab data $R = 10$, and the verification problem with lab data had a value of $R = 5$. For verification, each transmission is assigned to either the positive or negative class, and the ROC was calculated by thresholding the values calculated using Equation 4.11. The area under the curve (AUC) was calculated for the ROC, and the model with the highest AUC was tracked. Training was terminated when the AUC did not improve for R consecutive passes through the validation set, and the best performing model was saved.

4.2.3.3 Testing

During testing, the remaining 10% of transmissions were split into segments of 1,024 samples and given a forward pass through the network and the metric given in Equation 4.11 was calculated for each transmission. For identification, these values were compared, and the device with the maximum value was assigned as the class label. For verification, these values were thresholded to produce ROC and P-R curves for each verification network.

Actual Device \ Predicted Device	1	2	3	4	5	6	7
1	85	2	1	0	0	0	1
2	0	93	2	0	0	0	9
3	1	8	92	0	0	0	0
4	0	3	0	103	0	0	0
5	0	0	2	0	86	14	2
6	1	0	0	0	3	87	0
7	0	0	0	5	0	0	100

Table 4.4: The confusion matrix for the seven device identification problem for the outdoor data indicating the number of transmissions in each class of the test set and their classification.

4.2.3.4 Implementation

A NVIDIA GeForce GTX 1080 Ti GPU was used to train and test the network. All data pre-processing was done in MATLAB R2017a and the CNN was implemented in Python using Keras 2.0.9 [97] with TensorFlow 1.3.0 [98] as the backend.

4.3 Results and Discussion

Results from the outdoor and lab experiments are given in Figures 4.1-4.2 and Tables 4.4 - 4.8 . For both experiments, overfitting was observed during the late stages of training. This is evident because early on, the classification accuracy of the CNN on the training set approximately matched that of the validation set, while near the end of training the validation accuracy peaked, and the training accuracy continued to improve. For this reason, we track the model with the best performance on the validation set, and end training after the validation performance metrics stop improving as defined in Section 4.2.3.

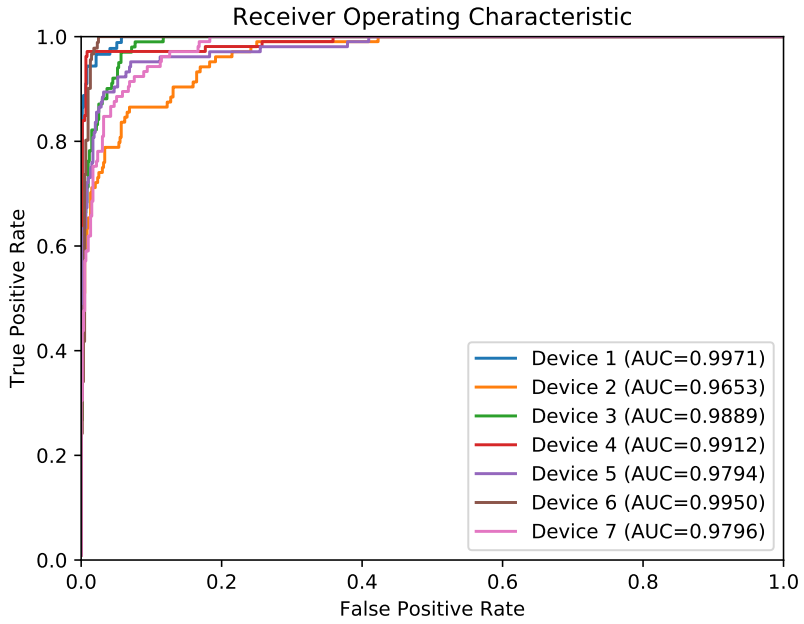


Figure 4.1: Receiver operating characteristic curves for each of the seven verification problems for the outdoor data. The AUC is included in the legend for each device.

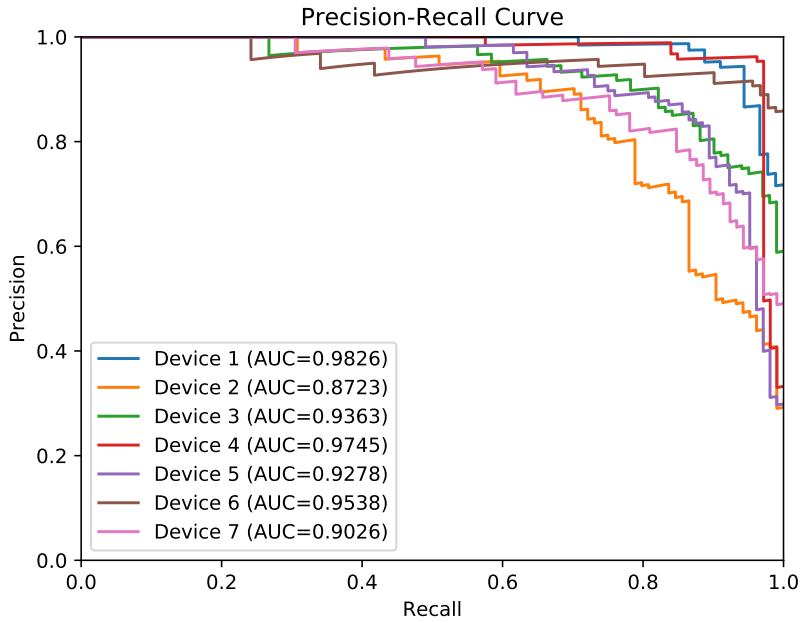


Figure 4.2: Precision-recall curves for each for the seven verification problems for the outdoor data, with the AUC for each curve given in the legend.

Network	Epochs	Training Time (minutes)
Identification	65	24.29
Verification 1	41	12.87
Verification 2	29	9.89
Verification 3	45	14.06
Verification 4	28	9.60
Verification 5	55	16.36
Verification 6	14	6.10
Verification 7	65	19.25

Table 4.5: The number of epochs until the best model was found and the total training time for each network for the outdoor data. (Note that training continued after the best model was obtained and this is included in the total training time.)

Network	Epochs	Training Time (minutes)
Identification	17	482.14
Verification 1	1	87.68
Verification 2	3	114.03
Verification 3	1	87.30
Verification 4	3	118.82
Verification 5	2	100.19
Verification 6	1	89.52
Verification 7	1	87.89

Table 4.6: The number of epochs until the best model was found and the total training time for each network for the lab data. (Note that training continued after the best model was obtained and this is included in the total training time.)

SNR (dB)	Accuracy (%)
10	73.73
15	85.45
20	89.92
25	90.61
30	91.13
35	91.38
40	91.38

Table 4.7: The overall accuracy at each SNR for the lab experiment identification problem.

SNR (dB)	ROC AUC	Precision-Recall AUC
10	0.9270	0.6935
15	0.9594	0.8213
20	0.9716	0.8820
25	0.9757	0.9016
30	0.9770	0.9070
35	0.9774	0.9092
40	0.9775	0.9103

Table 4.8: The average ROC AUC and average P-R AUC at each SNR for the lab experiment verification problem.

4.3.1 Outdoor Experiment

A confusion matrix for the identification problem is shown in Table 4.4. The overall correct identification rate is 92.29% and the correct classification rate over all 1,024 sample segments in the test set is 77.01%. This demonstrates that the metric given in Equation 4.11 combines the results of the 1,024-sample segments in a way that boosts the classification performance over that of a single segment. Results for the verification problem are given in Figures 4.1 and 4.2. The mean ROC AUC of all devices is 0.9852 indicating very strong verification performance. Additional information regarding the amount of training for each network is given in Table 4.5.

4.3.2 Lab Experiment

Table 4.6 describes the number of epochs of training until the best network parameters were found, as measured by the performance on the validation set, as well as the total training time in minutes for each network. It is important to note that since the network and training dataset in the lab experiment were larger, the total training time increases even though the network is able to converge in fewer epochs. Table 4.7 displays the overall accuracy on the test set at each SNR for the identification problem. As expected, the network performs better on higher SNR transmissions, with significant performance drop-off below 20 dB. Table 4.8 shows the average over all devices of the ROC AUC and average P-R AUC at different SNRs for the verification networks.

4.4 Conclusion

We have demonstrated that CNN techniques which provide state-of-the-art performance in image and speech recognition problems can be applied to the problem of RF fingerprinting. Our method relies on steady state analysis of the signal and achieves high identification and verification accuracy on seven ZigBee devices. Our method also permits use of the full transmission for fingerprinting regardless of the data content of the signal, rather than relying only on preambles or otherwise common bit sequences.

Furthermore, we have demonstrated that deep learning methods are suitable for RF fingerprinting of the types of devices used in IoT networks because of their ability to adapt to and discover the features available in the target signals that are useful for classification. In contrast, approaches based on models or features that are defined a priori may have difficulty detecting sufficient differences to distinguish the devices if those particular features are not the most prominent in the selected class of devices.

It is significant to note that the network used for lab data with multiple SNRs requires more parameters than the one used for outdoor data at effectively a fixed SNR. This demonstrates that increasing the size of the network and providing additional training data allows the network to learn additional channel effects while maintaining high performance. This shows promise for the ability of our technique to adapt to data collected under varying conditions, such as power levels and center frequencies, by training on a more diverse dataset and using a larger network.

Chapter 5: Enhancing RF Fingerprint Verification with Recurrent Neural Networks

5.1 Introduction

The methods presented in Chapter 4 provided promise that neural network techniques for RF fingerprinting are possible. In this chapter, we build upon the techniques provided in Chapter 4 and perform several experiments which evaluate the usefulness of such methods in real-world environments. First, we enhance the algorithm presented in Chapter 4 with an improved pre-processing sequence and the addition of RNN components to improve performance, particularly at low SNRs. Then, we measure the performance of our technique against devices that were not seen during the training process to more closely model performance in a real-world scenario. We evaluate this performance as a function of the total number of devices seen during training to show that the generalizability of the classifier is improved as the size of the population of known devices increases.

Next, we apply a simulated realistic multipath channel model and assess the classifier performance both with and without a priori knowledge of the distribution of multipath environments. Ren *et al.* have shown previously that the sort of tame

indoor office environments where such IoT devices would typically operate cause some degradation in fingerprint classification performance on a set of hand-crafted features, but that the effect is relatively limited and varies significantly amongst the different features, with transient features impacted most drastically [99]. Below, we perform a similar experiment, except we use a neural network classifier model and determine whether a priori knowledge of the distribution of multipath environments helps to mitigate the performance degradation.

Additionally, we emphasize the importance of frequency synchronization through a simple experiment that outlines the ease of fooling a classifier trained on unsynchronized transmissions. Finally, we present a simple technique for reducing the storage requirements of the trained models by 95% without harming classification performance.

5.2 Methods

5.2.1 Data Collection

During the data collection process, 6,000 transmissions were recorded at a sampling frequency of 16 MHz from each of 30 Digi XBP24CZ7SITB003 devices. Each DUT was connected to a Pulse Electronics W1030 omni-directional antenna inside a Ramsey Electronics STE3000B shielded RF test enclosure and configured as a ZigBee router transmitting at the maximum power level of 18 dBm using the Digi XCTU software [93]. Another identical antenna was placed inside the enclosure and wired outside to act as the receive antenna for that device. Each DUT was networked

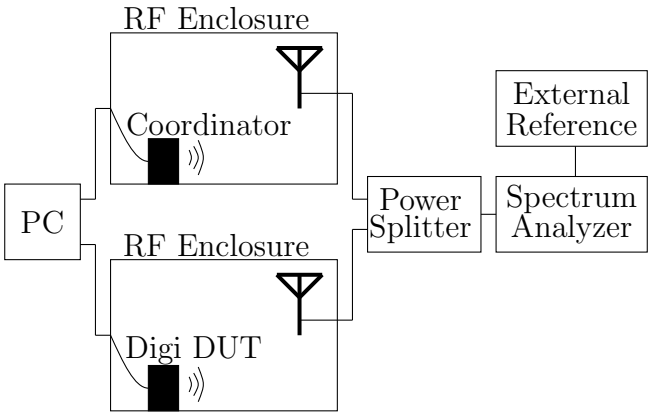


Figure 5.1: The data collection setup for the Digi devices.

one at a time with a separate device, the ZigBee coordinator, inside a separate enclosure in an identical configuration. Both receive antennas were connected to a power splitter, and the combined time-series signal was downconverted to baseband and saved on a Rohde & Schwarz FSW67 signal and spectrum analyzer with external reference oscillator. Each packet transmitted by the DUT was a 32 byte uniform randomly generated bit sequence with a two byte checksum, and was generated on a PC and transmitted to the DUT over USB. A diagram of this setup is shown in Figure 5.1.

5.2.2 Data Pre-processing

The data pre-processing sequence is similar to the steps followed in Chapter 4.2.2 since it outlines a method to generate a dataset free from certain biases, e.g. minor power differences or center frequency offsets amongst transmitters that could serve as trivial, unreliable, and easy-to-spoof features. The primary pre-processing differences in this chapter are the inclusion of randomly generated simulated mul-

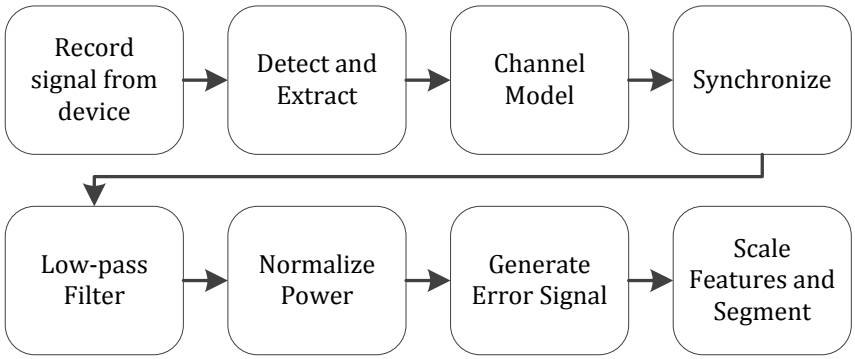


Figure 5.2: The sequence of pre-processing steps to prepare a transmission for classification by the neural network.

tipath environments, and the presentation of the entire signal to the network at once, rather than presenting only short, fixed-size signal segments and manually combining those estimates to create an overall prediction. In addition, a different synchronization technique is used as it is more robust at lower SNRs than the previous technique, and a wider range of SNRs are evaluated. The full data pre-processing chain is described below and summarized in Figure 5.2.

The first step is to detect and extract the packets transmitted by each DUT within the raw signal recordings. Since the data collection was performed inside of a sealed enclosure and has very high SNR, this is performed by simply thresholding the amplitude of the recorded signal. To measure the performance of our method in the presence of noisy multipath channels, each transmission is optionally passed through a unique, randomly generated simulated multipath channel described in Section 5.2.2.2, then combined with AWGN. Note that the simulated SNRs assume that each measured signal is recorded completely noise-free.

Each signal is then synchronized in frequency, phase, and time using the technique described in Section 5.2.2.1.

Because the data is recorded at 16 samples per symbol, we low-pass filter the signal to a 2 MHz passband using a FIR filter of order 151 designed using the Parks-McClellan algorithm [100]. In practice, this limits the ability of the neural network to consider out-of-band emissions as features. Despite this limitation, in a real-world setting, it is desirable to rely only on the in-band portion of the signal since other devices may be transmitting on adjacent channels.

The signals are power-normalized, then the estimated ideal signal is subtracted. The estimated ideal signal is produced simply be demodulating the received waveform, then modulating the received symbols at the same sample rate. The resulting signal is scaled by subtracting the mean of the signal and dividing by the standard deviation, performed separately for real and imaginary components. Finally, the signals are segmented by shaping them into a series of timesteps of 1,024 samples.

Within the neural network, the convolutional filters act separately on each segment of the signal, then recurrent layers operate across segments to combine information for the entire transmission. While a purely convolutional network does not adapt as well to signals of varying lengths and presents computational issues for extremely long signals, a purely recurrent network may have poor representation ability due to having few input features per timestep, and may be inefficient due to the large number of timesteps. Thus, we designed our network to utilize both convolutional and recurrent components to combine the benefits of both approaches.

5.2.2.1 Carrier Synchronization

We synchronize each individual transmission independently in phase, frequency, and time to the receiver. We follow the data-aided Rife and Boorstyn algorithm using 2^{18} FFT bins for coarse frequency synchronization outlined in [101]. Note that the data-aided approach requires an estimate of the transmitted signal, for which we use the estimated ideal signal that was already generated. We then perform fine synchronization using Newton's method as outlined in [102]. Finally, we perform timing synchronization by interpolating the signal by a factor of ten, aligning it via cross-correlation to the estimated ideal signal reproduced at ten times the sampling frequency, then downsampling the result back to the original rate.

5.2.2.2 Multipath Simulation

The distribution of simulated multipath channels was designed to reflect typical indoor conditions in which IEEE 802.15.4 devices are most likely to operate. Each time a transmission was added to a dataset for a given network at a particular SNR, a different multipath channel was randomly generated and its effects were imposed on the signal. In doing so, we aim to train the network to ignore the effects of any one particular channel and attain a model which is useful over a range of environments. Since each transmission at each SNR operates in a different channel, we demonstrate the ability of our technique to learn the patterns present in the signal itself, while ignoring channels effects whenever possible.

The number of path delays was selected from a uniform random distribution

on the interval $[1, 4]$, where one is equivalent to having a single transmission path between transmitter and receiver, and four implies three additional paths along which the signal is received. The distance d, in meters, between the transmitter and the receiver was chosen randomly from a uniform distribution on the interval $[10, 100]$. For each path, there is a corresponding time delay that represents the amount of time in nanoseconds that the delayed signal arrives at the receiver after the signal along the first path. Each delay was chosen from a uniform random distribution between 1 and $\min\left\{\frac{d}{300\times10^{-3}}, 100\right\}$ ns. This assumes that each path is no more than $d+30$ when $d \in [30, 100]$ m and no more than $2d$ when $d \in [10, 30]$ m, assuming the signal propagates at 300×10^6 m/s. By convention, the first path has a time delay of 0 ns. The gain in dB corresponding to each path delay n, is then calculated by the following equation

$$\text{gain}(n) = -(K + 10m \cdot \log\left(d + d'(n)\right) + \sigma r),$$

where $K = 53.7$, $m = 3.4$, $d'(n)$ is the difference between the distance that the path of delay n traveled to the receiver and d, $\sigma = 5.3$, and r follows a standard normal distribution [103]. This relationship between distance and gain is derived from a set of experiments measuring IEEE 802.15.4 transmitters operating in the 2.4 GHz band in indoor environments [103].

The generated path delays and path loss were then used as inputs to a Rayleigh channel using MATLAB's communications toolbox [104] which models the multi-path channel as a linear FIR filter. Because the signals are narrowband relative

to the delay spread of the channel, these channels are generally characterized as frequency flat as they typically have no more than a few dB of frequency-selective attenuation in the signal passband. While this may not significantly distort the signal for demodulation purposes, it was previously unclear how much effect this would have on RF fingerprinting since the differences between devices are already so small.

5.2.2.3 Dataset Generation

For each instance of the verification network, we select a different device to be the positive class, and place transmissions from the remaining devices in the negative class. The datasets are split so that 4,800 transmissions from each device are reserved for training, with another 600 reserved for each of the validation and testing sets. When constructing the datasets, each transmission is included in a given set for each of the following SNR levels in dB: {0, 5, 10, 15, 20, 25, 30, 35, and 40}. Note that the SNRs are computed for the oversampled signal prior to filtering, so the in-band SNRs are higher. For any given network, the classes are balanced so that the total number of transmissions in the negative class sets is equal to those of the corresponding positive class by randomly selecting a subset of transmissions from the negative class.

5.2.3 Neural Networks

We train verification networks to authenticate the identity of a single transmitting device against a population of other identical devices. The proposed network architecture is given in Table 5.1. The ELU activation function [94] was again used for all convolution and dense layers except at the output layer where sigmoid was used.

Layer	**Layer Description**	**Parameters**
Input	$17 \times 1024 \times 2$	-
Time-Distributed Convolution 1D	32×19	$1,248$
Time-Distributed Max Pooling	Pool size $= 2$	-
Time-Distributed Convolution 1D	128×19	$77,952$
Time-Distributed Max Pooling	Pool size $= 2$	-
Time-Distributed Convolution 1D	32×15	$61,472$
Time-Distributed Max Pooling	Pool size $= 2$	-
Time-Distributed Convolution 1D	16×11	$5,648$
Time-Distributed Max Pooling	Pool size $= 2$	-
Time-Distributed Flatten	-	-
Time-Distributed Dense	128 nodes	$106,624$
Time-Distributed Dropout	$p = 0.5$	-
Time-Distributed Dense	16 nodes	$2,064$
Time-Distributed Dropout	$p = 0.5$	-
LSTM	24 nodes	$3,936$
Dense	1 node	25

Table 5.1: The network architecture and parameter count of each layer.

All experiments were performed using four NVIDIA GeForce GTX 1080 Ti GPUs using Keras [97] with the Tensorflow backend [98]. The networks were trained using the Adam optimizer [96] with default parameters, a batch size of 1,024 transmissions, and with binary cross-entropy as the loss function. For each experiment, we first train an alternate network for each device for three epochs. The alternate

network is obtained by removing the LSTM layer from the architecture given in Table 5.1 and treating each signal segment as a unique training sample with its own device label. Next, the LSTM layer is inserted and the the remaining weights are initialized to the values from the trained alternate network.

The time-distributed layers in Table 5.1 indicate that the layer runs separately on each signal segment. This is illustrated in Figure 5.3, with the architecture of a CNN block given in Figure 5.4. The parameters in the CNN block are the same for all segments in a given verification network. Training of the network continues for up to 30 epochs, with early stopping if the area under the ROC curve evaluated on the validation set does not increase for ten consecutive epochs. At that time, the best-performing network throughout training is saved as the verification network for that device.

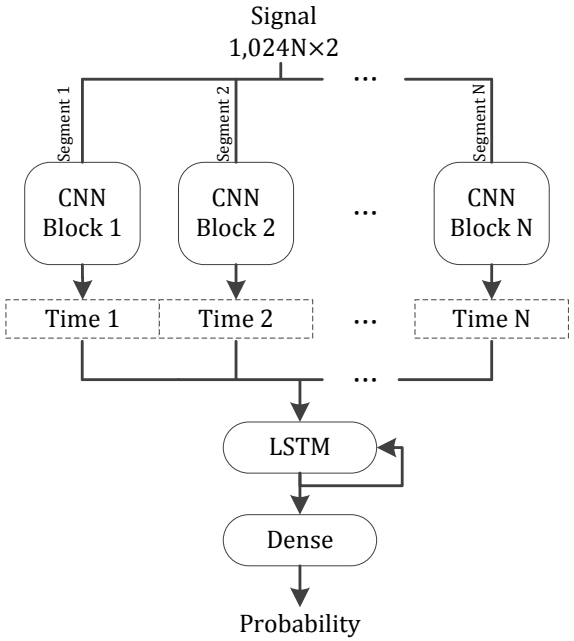


Figure 5.3: The overall neural network architecture.

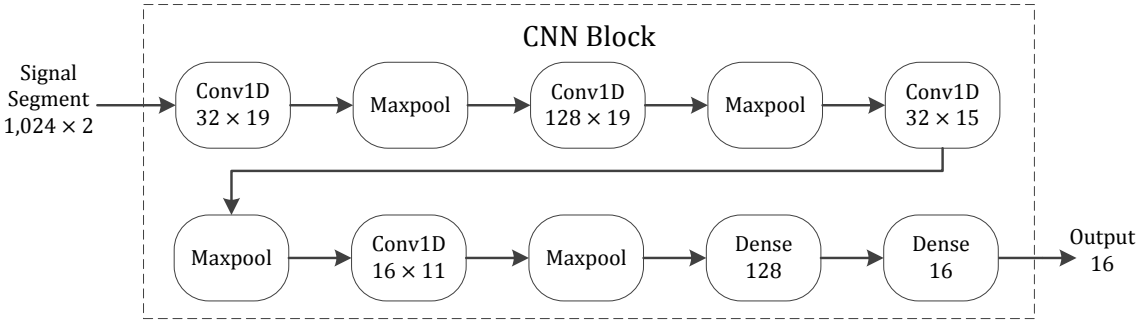


Figure 5.4: The architecture of a CNN block.

5.3 Experiments

This section provides an overview of each experiment performed using the updated verification network architecture and training algorithm.

5.3.1 Baseline

To begin, we train 25 different verification networks on 25 devices, where for each network there is a single device in the positive class, and the remaining 24 devices comprise the negative class. The baseline experiment simply evaluates each of these trained networks on a testing set comprised of new transmissions from the 25 devices. These devices are referred to as *known devices* since their transmissions are present in the training dataset. This experiment evaluates the ability of the classifier to distinguish transmissions from transmitters that it has previously seen. While less applicable to a realistic environment where devices from outside of the training set are being used, this test serves to indicate how well the classifier is able to learn the variability in a set of transmitters.

5.3.2 Withheld Devices

A more challenging problem is for the algorithm performance to generalize to devices that were not seen during training. Unfortunately, RF fingerprinting methods have limited practical use if they cannot generalize to previously unknown devices, since an unauthorized user is unlikely to use a known device to attempt to gain unauthorized network access. For this experiment, we evaluate the previously trained networks from the baseline experiment on *withheld devices* by replacing the negative class data from the baseline test with transmissions from the five remaining devices that have no transmissions included in the training sets. In addition, we expand upon this experiment by repeating the baseline training and withheld device evaluation procedure using a variable number of known devices in the negative class during training. This experiment tests the hypothesis that a negative class constructed with a larger number of known devices will generalize better, as evidenced by higher performance on the set of withheld devices.

5.3.3 Multipath Model

Next, we evaluate the impact that the simulated multipath model has on the ability of the algorithm to correctly classify the transmitting device. To do so, we perform two experiments. First, we evaluate the previously trained baseline model on a test dataset where the transmissions were passed through the multipath model. Next, we retrain the networks, replacing the training dataset with data that has passed through the simulated multipath model, and reevaluate the performance

on the multipath testing dataset. The first experiment will measure the impact that the multipath environment has on the classifier when it is not accounted for. The second experiment will assess whether the classifier can improve its results when the distribution of multipath environments is known a priori and learned as part of the training sequence.

5.3.4 Frequency-Shifted Signals

So far, we have synchronized the data prior to presentation to the network so that the network would not learn device-specific center frequency offsets. This was based on the assumption that an attacker could easily force a frequency-shift in their device, serving as an easy way to fool a classifier that relies on such center frequency offsets. For this experiment, we seek to validate that assumption. Here, we do not synchronize the received signals prior to training the classifier. Since the signals are not synchronized, the error signal is less meaningful, so the ideal signal is not subtracted from the measured signal.

Next, we evaluate the performance of the classifier against other unsynchronized transmissions from the set of known devices to measure how well these classifiers can work against a user that simply uses another device. Finally, we evaluate the classifier against transmissions from the withheld devices that were frequency-shifted in software to the mean frequency offset of the particular known device for that verification network. This simulates the effect of an unauthorized user purchasing another device, measuring the frequency offset in the received transmissions

from the trusted device, and shifting the center frequency of their device to try to match that of the trusted device. Such an attack would clearly fail against a classifier which synchronizes the transmissions first.

If a signal $x_{\text{meas}}^{a,i}(t)$, corresponding to the i^{th} transmission from device a, could be synchronized in frequency and phase with parameters $\omega^{a,i}$ and $\theta^{a,i}$ as follows

$$x_{\text{sync}}^{a,i}(t) = x_{\text{meas}}^{a,i}(t)e^{-j(\omega^{a,i}t+\theta^{a,i})}, \tag{5.1}$$

then the signal $x_{\text{shift}}^{b,a,i}(t)$, frequency-shifted to have the same frequency error as device b, is given by

$$x_{\text{shift}}^{b,a,i}(t) = x_{\text{meas}}^{a,i}(t)e^{-j((\omega^{a,i}-\bar{\omega}^{b})t+\theta_{\text{rand}})}, \tag{5.2}$$

where θ_{rand} is a uniform random variable on $[-\pi, \pi]$ due to the lack of phase synchronization and $\bar{\omega}^{b}$ is the mean frequency offset computed over all transmissions in the training set of device b, given by

$$\bar{\omega}^{b} = \frac{1}{M}\sum_{i=1}^{M}\omega^{b,i}, \tag{5.3}$$

where M is the number of transmissions in the training set belonging to device b.

5.3.5 Reduced-Memory Model

Finally, we seek to reduce the memory footprint of the trained models so that IoT devices, which are typically inexpensive and low-power, can store a larger

number of verification networks in their memory. In the naive approach, each trained model would be individually stored in a device's memory, requiring storage of 258,969 unique parameters per model, despite every network having an identical architecture. One approach to addressing this problem is to reuse as many parameters as possible from one network to the next via a transfer learning approach. To achieve this, we retrain the networks all from the same initialization and disable learning on selected parameters.

To begin, we select the best performing baseline model and initialize all networks to that model. Next, learning is disabled on the 128×19 and 32×15 convolution layers and the 128-node dense layer, forcing each network to use the exact same values for those parameters. If there are D devices, this reduces the memory required from $258,969 \times D$ parameters to $258,969 + 12,921 \times (D-1)$ parameters, a 95% reduction in the limit as $D \to \infty$. The motivation for this approach is that since all verification networks are looking for similar sorts of patterns, it is likely that the remaining free parameters within each model are enough to learn unique features for each device.

5.4 Results and Discussion

The results of the experiments are shown in Table 5.2, where the mean ROC and P-R AUC is reported at each SNR for each experiment, averaged across all verification networks. As a comparison metric, we measure the distance of a classifier from the ideal classifier as $1 - \text{AUC}$, where 0 represents the ideal value and

1 corresponds to a classifier that is always incorrect. As compared to technique A, technique B is said to be $G\%$ closer to an ideal classifier, where

$$G = 100 \times (1 - \frac{1 - \text{AUC}_B}{1 - \text{AUC}_A}). \tag{5.4}$$

For the baseline test, the performance greatly exceeds the results presented in Chapter 4 as a result of the improved pre-processing sequence and the inclusion of the LSTM layer to consider the entire signal at once, instead of only local information in a single segment. Previously, verification networks with only seven devices demonstrated P-R AUC of up to 0.9103 at high SNR, compared to 0.9713 demonstrated here, a result that is 68% closer to the ideal classifier despite a significantly larger population of devices. In addition, those results show less noise tolerance, with only 0.8213 P-R AUC at 15 dB SNR, which is surpassed even by the 0 dB SNR P-R AUC, 0.8354, shown here.

The trained networks clearly perform better when evaluated on the set of known devices than the set of withheld devices, with P-R AUC that is 8.5% closer to ideal at 0 dB SNR and 43.61% closer to ideal at 40 dB SNR. These results are not surprising, as neural networks tend to perform best when presented with data from a similar distribution to which they were trained on. Still, the results on the withheld devices dataset show great promise for applying these techniques in the real world to authenticate a single known device against a population of unknown devices, with 0.9651 ROC AUC at 40 dB SNR.

Experiment	Metric	SNR (dB)								
		0	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**
Baseline	Mean ROC AUC	0.8627	0.9343	0.9663	0.9772	0.9800	0.9800	0.9810	0.9805	0.9798
	Mean P-R AUC	0.8354	0.9144	0.9546	0.9671	0.9718	0.9729	0.9731	0.9726	0.9713
Withheld Devices	Mean ROC AUC	0.8449	0.9165	0.9495	0.9579	0.9619	0.9636	0.9625	0.9655	0.9651
	Mean P-R AUC	0.8201	0.8908	0.9313	0.9403	0.9441	0.9470	0.9446	0.9497	0.9491
Train without Multipath, Test with Multipath	Mean ROC AUC	0.8488	0.9049	0.9305	0.9371	0.9389	0.9399	0.9381	0.9384	0.9355
	Mean P-R AUC	0.8178	0.8782	0.9125	0.9201	0.9256	0.9277	0.9244	0.9245	0.9206
Train with Multipath, Test with Multipath	Mean ROC AUC	0.8651	0.9314	0.9590	0.9684	0.9704	0.9708	0.9717	0.9714	0.9703
	Mean P-R AUC	0.8410	0.9147	0.9470	0.9606	0.9634	0.9628	0.9634	0.9639	0.9611
No synchronization	Mean ROC AUC	0.9768	0.9821	0.9838	0.9847	0.9836	0.9848	0.9844	0.9851	0.9847
	Mean P-R AUC	0.9681	0.9755	0.9774	0.9783	0.9769	0.9792	0.9780	0.9794	0.9787
Train without synchronization, Test with frequency shift	Mean ROC AUC	0.6464	0.6825	0.6989	0.7080	0.7104	0.7083	0.7076	0.7077	0.7080
	Mean P-R AUC	0.6512	0.6908	0.7054	0.7194	0.7206	0.7185	0.7188	0.7167	0.7188
Reduced Memory Baseline	Mean ROC AUC	0.9027	0.9615	0.9821	0.9897	0.9910	0.9921	0.9912	0.9915	0.9910
	Mean P-R AUC	0.8777	0.9491	0.9764	0.9867	0.9875	0.9896	0.9880	0.9884	0.9877
Reduced Memory Withheld Devices	Mean ROC AUC	0.8883	0.9372	0.9589	0.9641	0.9665	0.9666	0.9656	0.9668	0.9657
	Mean P-R AUC	0.8654	0.9129	0.9406	0.9449	0.9498	0.9490	0.9479	0.9482	0.9483

Table 5.2: The results of the experiments described in Section 5.3.

Similar results are obtained in the case of multipath, where classifiers trained on signals without multipath effects perform substantially worse against signals with simulated multipath effects than classifiers that included the multipath effects in the training data. In this experiment, we measure a 12.73% improvement in P-R AUC at 0 dB SNR and a 51.01% improvement at 40 dB SNR when including the multipath effects in the set of signals used for training. Of course, the results are still not quite as strong as the baseline classifier on signals recorded without multipath effects. Still, this demonstrates that, as expected, the relatively tame multipath effects experienced by these narrowband signals cause only minor degradation in classifier performance, as long as they are accounted for during the training process.

The results of the synchronization test confirm our hypothesis that networks trained on unsynchronized transmissions will depend on the frequency error in each device as the primary feature for discrimination. These verification networks perform extremely well when evaluated against transmissions that are also unsynchronized, with performance at low SNR far surpassing that of the baseline test. This is to be expected, since a frequency offset in the baseband signal would persist at low SNR. However, the performance decreases drastically when the transmissions in the test set are frequency-shifted to match the positive class device, as given by Equation 5.2, with ROC AUC of only 0.7080 at 40 dB SNR and with 6/25 verification networks performing worse than a classifier that guesses randomly.

Surprisingly, the results of the reduced memory model show improvement over the baseline model in both the set of known and withheld devices. This is likely due to the increase in total training time for the reduced memory model since the model

with which it was initialized had already been through the entire training sequence. This demonstrates that despite reducing the number of trainable parameters in the model to only 5% of its original value, the network performance remains high.

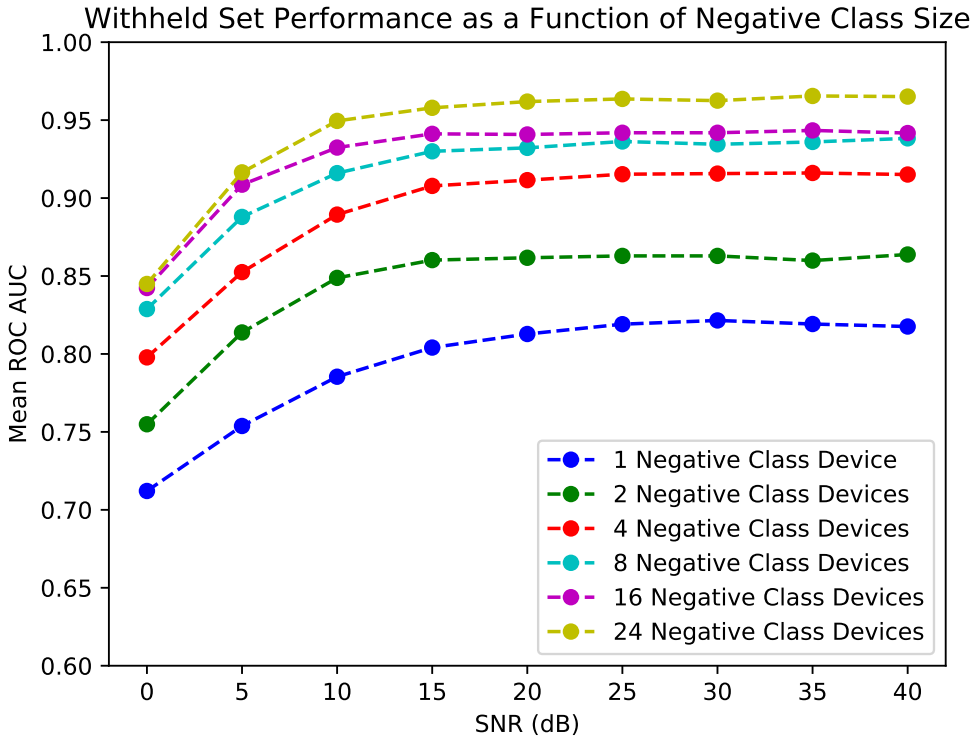
Finally, the results of the experiment comparing the performance on the withheld devices dataset as a function of the number of devices in the negative training class are shown in Figure 5.5. From these plots, it is clear that the ability of the verification network to generalize and properly assign transmissions from new devices to the negative class is directly related to the number of negative class devices seen during training. This is intuitive, since the probability that a new device is more similar to a negative class device than the positive class device increases with the number of devices in the negative class.

5.5 Conclusion

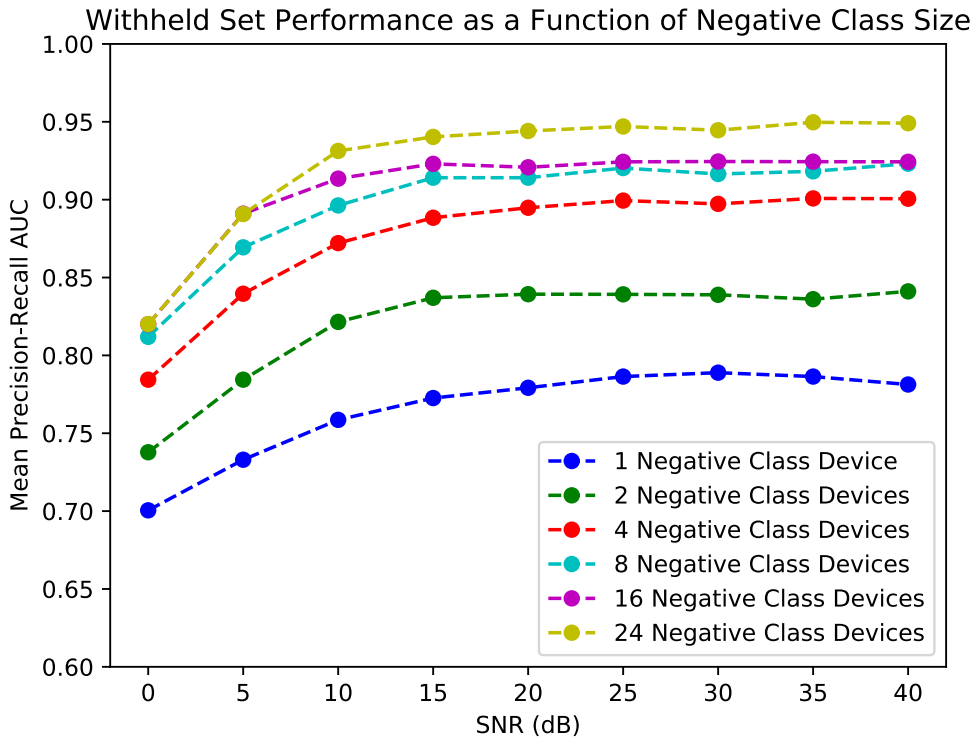
We have demonstrated a series of enhancements for neural network techniques for RF fingerprinting, particularly those that are geared towards IoT applications. The enhancements to the pre-processing sequence as well as the addition of recurrent components to the neural network model show significantly improved performance over the technique presented in Chapter 4, particularly at low SNRs.

Furthermore, we have shown through experiments several important factors which govern the performance of verification networks. First, we showed that the performance can generalize to new devices provided that transmissions from a large set of similar devices are included in the negative class during training. In addition,

we showed that multipath effects from the sort of environment in which IoT devices are typically used has a relatively minor impact on the performance, but only if a similar set of multipath effects are seen in the transmissions during training. We also demonstrated the importance of frequency synchronization of transmissions prior to fingerprint extraction by showing that a network trained on unsynchronized transmissions becomes unreliable when presented with frequency-shifted signals. Finally, we demonstrated a simple technique for reducing the memory requirements of a library of trained models by up to 95% without any loss of performance.



(a) Receiver Operating Characteristic



(b) Precision-Recall

Figure 5.5: The (a) ROC and (b) P-R AUC on the set of withheld devices for each SNR as a function of the number of negative class devices included in the training set.

Chapter 6: Generative Adversarial Networks for Improved Security in IoT RF Fingerprinting Systems

6.1 Introduction

The neural network techniques presented in Chapters 4 and 5 work well and successfully defend against weak and intermediate adversaries due to the careful pre-processing sequence which removes the effect of high-level easily-spoofed features, such as absolute power level and exact center frequency. However, no effort has been made in either of these techniques to mitigate the strong adversary that has precise, sample-by-sample control over their output waveform. This is particularly relevant due to recent work in image processing which has introduced vulnerabilities in neural network models, namely, that by presenting them with carefully-crafted input signals, they can be made to predict the incorrect class with high confidence [105–107].

This chapter explores this vulnerability as it applies to the RF fingerprinting problem. First, the GAN framework introduced by Goodfellow *et al.* [105] is adapted for use within the RF fingerprinting context. This requires modifying the optimization problem to accommodate typical RF constraints, such as bandwidth,

power level, and bit error rate. Then, a GAN is trained to generate signals that appear similar to those from a trusted device, and we demonstrate that an existing RF fingerprint classifier can be deceived by signals generated by the GAN. We then show that by augmenting the training dataset of the classifier with samples produced by an independent GAN, this vulnerability can be mitigated. This work emphasizes the need for extensive training of RF fingerprinting algorithms with a diverse set of both real and simulated transmitters.

Despite the impressive performance of neural network techniques for RF fingerprinting, we are aware of only one other application of adversarial machine learning to the RF fingerprinting problem [108]. Our work is novel in that we modify the framework of data augmentation GANs (DAGANs) [109] to allow a non-random signal to be used as input to the generator network. Antoniou *et al.* presented the DAGAN architecture to derive modified versions of an input image to increase the amount of data available for training classifiers [109]. As opposed to standard GAN models, in which the generator takes only a random vector as input, their generator network requires an input image on which to impose changes. Our network model is structured similarly to this model so that a non-random input signal may be used.

Like many neural network techniques, GANs were initially applied in the computer vision domain, with applications such as generating new examples of handwritten digits and face images [6]. In follow-up work, Goodfellow *et al.* demonstrated that adversarial examples generated by the addition of small, imperceptible targeted noise to an input image, could force a high-confidence incorrect classification by a trained classifier [105]. Kurakin *et al.* later showed that the effect of these

seemingly imperceptible changes often persist even when the adversarial images are printed and recaptured via another camera [106].

6.2 Methods

The GAN formulation introduced by Goodfellow *et al.* [6] is based on a two-player minimax game where the generator seeks to minimize the function

$$\frac{1}{B}\sum_{k=1}^{B}\log\left(1-D\left(G\left(z^{k}\right)\right)\right), \tag{6.1}$$

while the discriminator seeks to maximize the function

$$\frac{1}{B}\sum_{k=1}^{B}\left[\log\left(D\left(y^{k}\right)\right)+\log\left(1-D\left(G\left(z^{k}\right)\right)\right)\right], \tag{6.2}$$

where B is the mini-batch size, y^k is the k^{th} sample in the training dataset, z^k is the k^{th} random vector, D is a function representing the discriminator, and G is a function representing the generator.

The RF fingerprinting problem is framed slightly differently, as we seek to generate not just any signal that could fool the discriminator, but only valid signals that could be demodulated correctly. The use of the DSSS technique in IEEE 802.15.4 devices presents additional challenges, as only a subset of the possible sequences of symbols are valid. Consequently, we develop a framework in which the generator takes an ideal signal, modulated precisely as defined in the standard, and imparts imperfections on the signal similar to those seen in the actual measured

signals from that device.

This motivates the use of a network similar to the DAGAN model [109] which generates transformed versions of an input image. However, their network first encodes the input signal in a lower-dimensional space before decoding back to the original dimension. In our case, this was found to produce signals that were drastically different from the input signal and failed to demodulate correctly. As a result, the dimensionality is maintained at each step in the generator network by avoiding downsampling operations and by zero-padding convolution operations. Additionally, the DAGAN network input image is a real image, while our algorithm first transforms the real signal through demodulation and re-modulation to recover the ideal signal. Such a transformation does not apply to image processing tasks where there is no notion of an ideal image that can be recovered.

Because no two transmissions from the same transmitter are identical even if the data contents match, a random vector input to the generator network provides a source of randomness from which to generate different outputs even for the same input signal.

Additional steps are taken in an attempt to minimize the differences between generated and measured signals. Both signals are low-pass filtered to ensure that only in-band effects are used for classification. Additionally, the signals are power normalized by dividing them by the RMS of the amplitude signal. Mathematically, we define function P as the cascade of these two processing steps. Finally, the ideal signal is subtracted from each signal to produce the error signal, which has been shown in Chapters 4 and 5 to be a useful signal for discriminating devices.

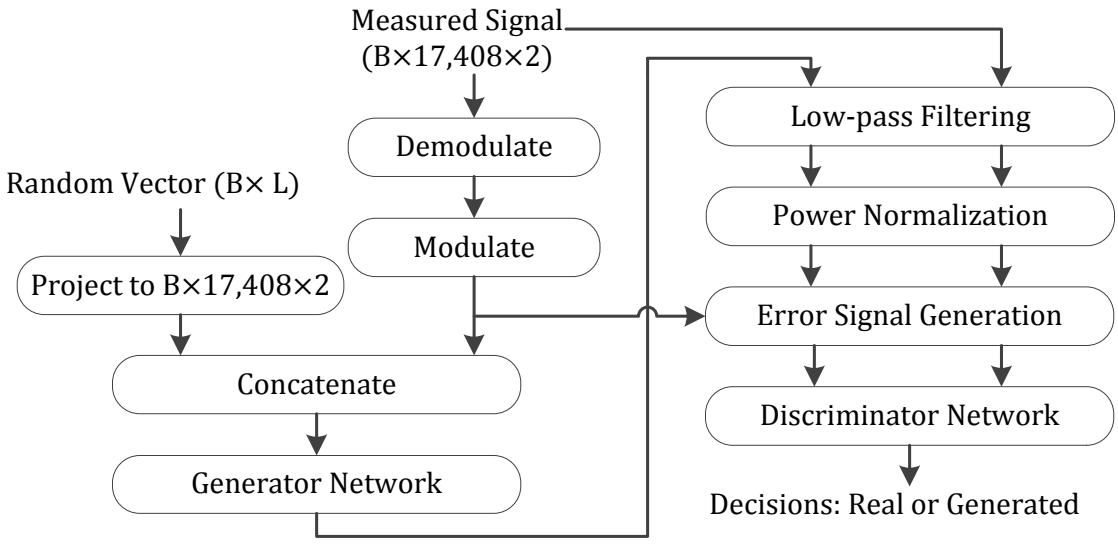


Figure 6.1: The architecture of the proposed GAN system.

To handle these additional complications, and with an update to the original cost functions to account for vanishing gradients [110], the optimization now seeks to minimize the following generator cost function

$$-\frac{1}{B}\sum_{k=1}^{B}\left[\log\left(D\left(P\left(G\left(x^k,z^k\right)\right)-x^k\right)\right)-\alpha\left\|P\left(G(x^k,z^k)\right)-x^k\right\|^2\right], \tag{6.3}$$

while alternately attempting to maximize the following discriminator cost function

$$\frac{1}{B}\sum_{k=1}^{B}\left[\log\left(D\left(P\left(y^k\right)-x^k\right)\right)+\log\left(1-D\left(P\left(G\left(x^k,z^k\right)\right)-x^k\right)\right)\right], \tag{6.4}$$

for some value of hyperparameter α which keeps the generated signals close enough to the ideal signal to allow error-free demodulation, and where x^k is the ideal input signal. The system architecture is shown in Figure 6.1, and the training algorithm is summarized in Algorithm 1.

At initialization, we aim for the distribution of values in the linearly projected

random vector to approximately follow a standard normal distribution. The required projection occurs by first computing $v = Wr$ for trainable matrix W of dimension $34,816 \times L$, and random vector r of length L, then reshaping v to dimension $17,408 \times 2$ so that it is the same size as the ideal modulated input signal. If all elements r_i of vector r and the initial values $w_{i,j}$ of matrix W are independently drawn from unif$(-a, a)$ for some value a, then

$$\mathbb{E}[v_i] = \mathbb{E}\left[\sum_{j=1}^{L} w_{i,j} r_j\right] = L\mathbb{E}[w_{i,j} r_j] = L\mathbb{E}[w_{i,j}]\mathbb{E}[r_j] = 0,$$

and

$$\mathrm{Var}[v_i] = L\mathrm{Var}[w_{i,j} r_j],$$

$$= L\left[\big[\mathbb{E}[w_{i,j}]\big]^2 \mathrm{Var}[r_j] + \big[\mathbb{E}[r_j]\big]^2 \mathrm{Var}[w_{i,j}] + \mathrm{Var}[w_{i,j}]\mathrm{Var}[r_j]\right],$$

$$= L\big[\mathrm{Var}[w_{i,j}]\mathrm{Var}[r_j]\big] = \frac{a^4 L}{9}.$$

$\mathrm{Var}[v_i] = 1$ requires that $a = \left(\frac{9}{L}\right)^{\frac{1}{4}}$. Since each v_i is the sum of L independent and identically distributed random variables, the Central Limit Theorem assures that the distribution of each v_i approaches the standard normal distribution as $L \to \infty$. In practice, we use $L = 10$ and $L = 20$, but verify through simulation that the distributions are well-approximated by the standard normal distribution.

Once the GAN is trained, a sample may be produced by the generator by modulating a random bit pattern into ideal signal x, drawing a random vector z

Input: A set of N transmissions $\{y^1, y^2, \ldots, y^N\}$ recorded from the same transmitter and synchronized to the receiver.

Algorithm:

for $i \leftarrow 1$ **to** N **do**

$x^i \leftarrow \text{modulate}\left(\text{demodulate}\left(y^i\right)\right)$

end

for $i \leftarrow 1$ **to** $3,000$ **do**

for $j \leftarrow 1$ **to** 3 **do**

- Sample mini-batch of input samples: $\{x^1, x^2, \ldots, x^B\}$
- Sample mini-batch of noise samples: $\{z^1, z^2, \ldots, z^B\}$
- Update generator parameters via the Adam algorithm for stochastic gradient descent on the function:

$$-\frac{1}{B}\sum_{k=1}^{B}\left[\log\left(D\left(P\left(G\left(x^k, z^k\right)\right) - x^k\right)\right) - \alpha\left\|P\left(G(x^k, z^k)\right) - x^k\right\|^2\right]$$

end

- Sample mini-batch of input samples: $\{x^1, x^2, \ldots, x^B\}$
- Sample mini-batch of corresponding received transmissions: $\{y^1, y^2, \ldots, y^B\}$
- Sample mini-batch of noise samples: $\{z^1, z^2, \ldots, z^B\}$
- Update discriminator parameters via the Adam algorithm for stochastic gradient descent on the function:

$$-\frac{1}{B}\sum_{k=1}^{B}\log\left(D\left(P\left(y^k\right) - x^k\right)\right) + \log\left(1 - D\left(P\left(G\left(x^k, z^k\right)\right) - x^k\right)\right)$$

end

Algorithm 6.1: The proposed GAN training algorithm.

from unif$(-a, a)$, and calculating $P(G(x, z))$.

To measure the effectiveness of this approach, we experimented with a set of 25 Digi XBP24CZ7SITB003 ZigBee PRO devices. From each device, 9,000 transmissions were downconverted to baseband and recorded at a sampling frequency of 16 MHz. We train individual verification networks for each device using 3,000 transmissions with random data contents for training, validation, and testing of each model using the architecture and technique described in Chapter 5. We will

Layer	Description	Activation	Parameters
Input	$B \times 17,408 \times 4$	-	-
Convolution 1D	64×32	Tanh	$8,256$
Convolution 1D	32×25	Tanh	$51,232$
Convolution 1D	4×19	Tanh	$2,436$
Convolution 1D	2×16	Tanh	130
Output	$B \times 17,408 \times 2$	-	-

Table 6.1: The generator network architecture for GAN1.

show that these networks, trained exclusively on transmissions from real devices, are insufficient at defending against adversarial examples produced by a GAN. We then update the training protocol to protect against this shortcoming.

Next, we use another 3,000 transmissions per device to train, validate, and test the proposed GAN algorithm using the generator and discriminator architectures outlined in Tables 6.1 and 6.2. A mini-batch size of 1,024 transmissions was used and training proceeded for 3,000 iterations, where an iteration includes alternately training the generator for three mini-batches and the discriminator for one mini-batch. Because the parameters of both the generator and discriminator are constantly being updated in an effort to improve the performance relative to the other, it is difficult to select an appropriate time to cease training and accept the generator as sufficiently trained. To account for this, the model parameters are saved every 50 iterations to allow the model to be reloaded at many different points throughout training. To match the technique used by the classifier, the measured transmissions are first synchronized to the receiver and trimmed so that only the random payload portion of the signal is preserved.

For each device and each of the generator models saved after the first 2,000

Layer	Description	Activation	Parameters
Input	$B \times 17,408 \times 2$	-	-
Reshape	$B \times 17 \times 1,024 \times 2$	-	-
Convolution 1D*	32×19	ELU	$1,248$
Average Pool*	Pool size $= 2$	-	-
Convolution 1D*	16×19	ELU	$9,744$
Average Pool*	Pool size $= 2$	-	-
Flatten*	$B \times 3,872$	-	-
Dense*	32 nodes	ELU	$123,936$
Dropout*	$p = 0.5$	-	-
Dense*	16 nodes	ELU	528
Dropout*	$p = 0.5$	-	-
Reshape	$B \times 17 \times 16$	-	-
LSTM	24 nodes	Tanh	$3,936$
Dense	1 node	Sigmoid	25
Output	$B \times 1$	-	-

Table 6.2: The discriminator network architecture for GAN1. Layers marked with an asterisk are performed individually on each of the 17 time segments.

iterations of training, we perform the following test. The verification network for the device is loaded, and the positive class from the original testing set for that network, corresponding to 300 real transmissions from that device at each SNR in $\{0, 5, \ldots, 40\}$ dB, is assigned as the positive class. For the negative class, 300 transmissions are generated by the trained generator and AWGN is added at the same SNRs as the positive class. All transmissions are low-pass filtered, power-normalized, and converted to error signals, then presented to the verification network. For each device, we save the results of the worst-performing model, corresponding to the stage of training during which the generator was most effective. Prior to evaluating each model, we generate and attempt to demodulate 300 transmissions from that model. If any bit errors are detected in the demodulated signals, that model is discarded and the results are not included.

Layer	Description	Activation	Parameters
Input	$B \times 17,408 \times 4$	-	-
Convolution 1D	64×25	Tanh	$6,464$
Convolution 1D	8×17	Tanh	$8,712$
Convolution 1D	2×9	Tanh	146
Output	$B \times 17,408 \times 2$	-	-

Table 6.3: The generator network architecture for GAN2.

Finally, we seek to counteract this attack method by augmenting the verification network training dataset with artificially-generated transmissions from the generator network. In a realistic scenario, the GAN used to augment the training dataset would not match the one used by the attacker. To model this difficulty, we train a second GAN per device using the architectures in Tables 6.3 and 6.4. This second GAN also uses different training data, different values of α, a different low-pass filter, a random vector input of length 10 instead of 20, and a different learning rate. Finally, the activation function on this GAN has been replaced with the leaky rectified linear unit (ReLU) [111].

Next, we reload the trained verification network for each device and replace half of the negative class samples from the training and validation datasets with signals output from the generator network, uniformly sampled from models at all stages of training in 50 iteration increments. We then fine-tune each of the verification networks for ten epochs, saving the model which performs best on the validation set. To demonstrate the benefit of augmenting the training procedure with a GAN, we measure the performance of the retrained networks against the original trained GANs. We will refer to the GANs used to evaluate the resilience of the trained verification networks as GAN1, and the GANs used to augment the training procedure

Layer	Description	Activation	Parameters
Input	$B \times 17,408 \times 2$	-	-
Reshape	$B \times 17 \times 1,024 \times 2$	-	-
Convolution 1D*	32×25	Leaky ReLU	$1,632$
Average Pool*	Pool size $= 2$	-	-
Convolution 1D*	16×17	Leaky ReLU	$8,720$
Average Pool*	Pool size $= 2$	-	-
Convolution 1D*	8×15	Leaky ReLU	$1,928$
Average Pool*	Pool size $= 2$	-	-
Flatten*	$B \times 912$	-	-
Dense*	64 nodes	Leaky ReLU	$58,432$
Dropout*	$p = 0.5$	-	-
Dense*	16 nodes	Leaky ReLU	$1,040$
Dropout*	$p = 0.5$	-	-
Dense*	4 nodes	Leaky ReLU	68
Dropout*	$p = 0.5$	-	-
Reshape	$B \times 68$	-	-
Dense	8 nodes	Leaky ReLU	552
Dense	1 node	Sigmoid	9
Output	$B \times 1$	-	-

Table 6.4: The discriminator network architecture for GAN2. Layers marked with an asterisk are performed individually on each of the 17 time segments.

as GAN2, where GAN1 and GAN2 are trained individually per device.

All GAN training and evaluation is performed using TensorFlow [98]. Optimization of the cost functions in Equations 6.3 and 6.4 is performed using the Adam optimizer [96] with learning rates of 10^{-3} for GAN1 and 10^{-4} for GAN2.

6.3 Results and Discussion

Figure 6.2 shows an ideal network input signal, the corresponding signal measured from a real device, and the signal produced by the trained GAN1. These images demonstrate that the GAN is able to transform the ideal input signal to one that looks much more similar to the signals actually produced by the target device.

Due to the random vector input to the generator, minor fluctuations occur in the exact signal output from one run to the next even for the same ideal input signal.

Figure 6.3(a) shows average ROC curves at each SNR for the verification networks trained exclusively on real transmissions, where the negative class corresponds to transmissions from GAN1. The AUC is included in the legend for each SNR. These ROC curves are computed by calculating the average value of the true positive rate (TPR) for each possible value of the false positive rate (FPR) across all devices. Since ROC curves inherently have jump discontinuities, we compute the minimum and maximum value of TPR for each possible FPR and average these values pointwise across all devices.

These curves indicate extremely poor performance at distinguishing real transmissions sent by the true positive class device from those created by GAN1, indicating a vulnerability of the verification network to this sort of attack. A classifier which blindly guesses may achieve a ROC curve where TPR = FPR, and AUC = 0.5. The curves generated here fall below that line for high values of FPR, and have AUC metrics which demonstrate extremely unreliable classification. The ROC AUC numbers for each individual verification network evaluated against GAN1 indicate that 10/25 networks perform worse than random guessing at 0 dB SNR, and 9/10 of those networks also perform worse than random guessing at 40 dB SNR. This emphasizes the need for methods to improve the robustness of existing RF fingerprinting techniques against such attacks.

Figure 6.3(b) shows average ROC curves for the verification networks when transmissions produced by GAN2 are used to augment the training dataset. These

ROC curves indicate extremely strong performance at distinguishing real transmissions produced by the target device from artificially generated signals. This experiment demonstrates that GAN1 is able to produce transmissions that are realistic enough to fool a neural network that has not been previously trained using a GAN, but not realistic enough that discrimination becomes impossible or even difficult if a GAN is used to augment training.

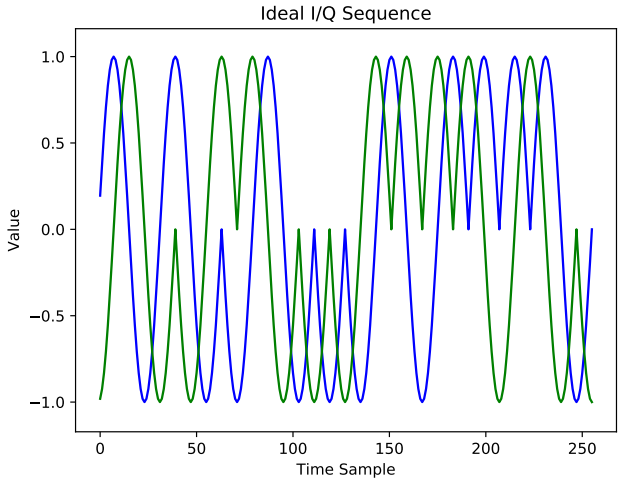
This result has important implications for neural network-based RF classification systems. While these signals are artificially generated and not produced by a real transmitter which would impose its own fingerprint on top of the desired signal, it demonstrates a vulnerability in the trained networks which could potentially be exploited by an attacker. The ability of the network to overcome this by training on adversarial examples emphasizes the need for a training dataset with as much diversity as possible to limit such vulnerabilities.

In particular, the ability of the GAN to learn to effectively fool the classifier without hands-on access to it is noteworthy and demonstrates a serious shortcoming in the training of these networks, one that is not apparent when performance is assessed on real transmitters. While the training dataset and architecture of the classifier and GAN discriminator are constructed similarly, the samples included in each training dataset are different for both networks. In addition, the network hyperparameters are unique for GAN1 and GAN2, and the parameters are initialized and learned independently. The ability to mitigate this vulnerability using a different GAN with a unique architecture, initialization, dataset, and hyperparameters illustrates the need for greater attention to training data diversity and use

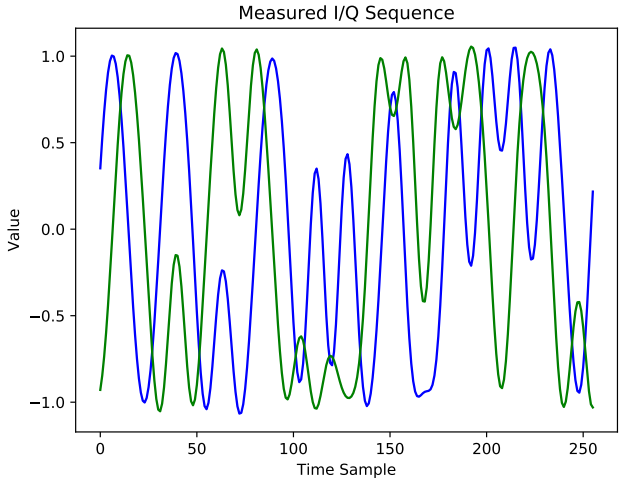
of adversarial examples in the training process for systems designed for security applications.

6.4 Conclusion

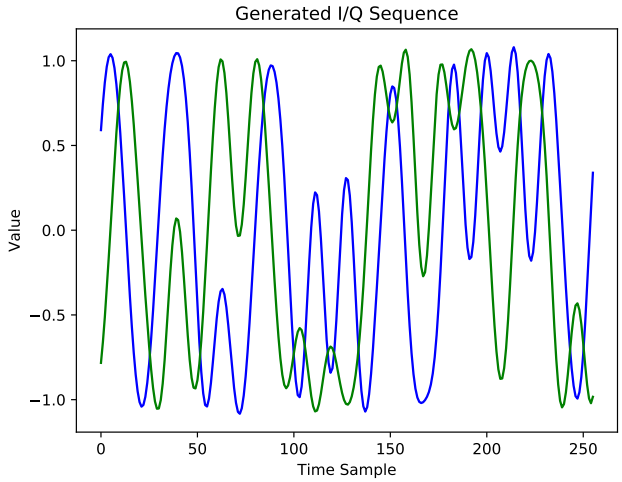
In this chapter, we have outlined how the GAN framework can be adapted for use with RF signals and demonstrated via experiment how neural networks for RF fingerprinting can be fooled by a GAN model. Furthermore, we have shown that augmenting the training dataset with adversarial examples can strengthen the neural network model against such attacks, effectively defending against a strong adversary making sample-by-sample changes to their transmitted waveform.



(a) Sample ideal signal

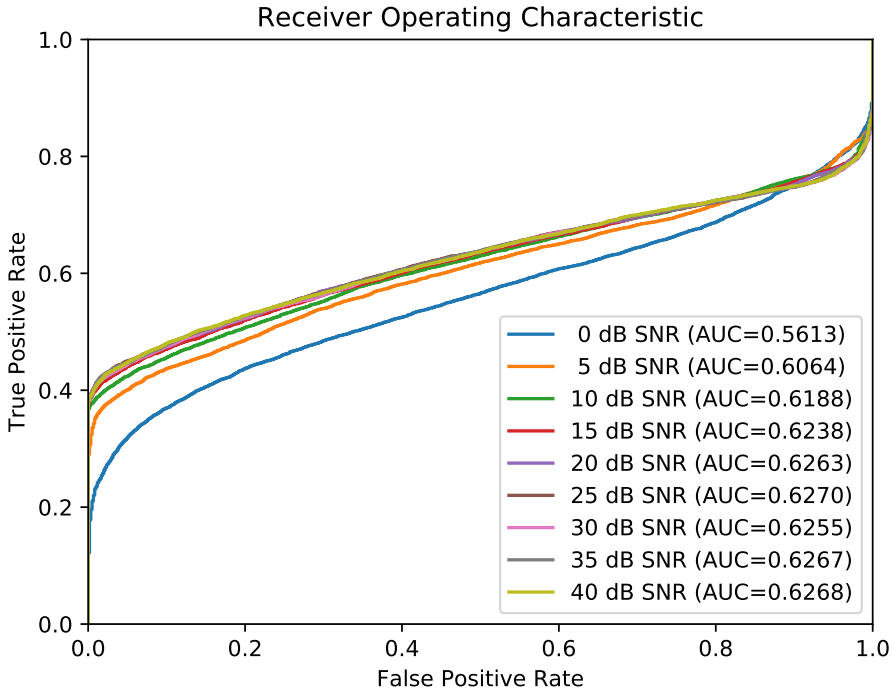


(b) Sample measured signal

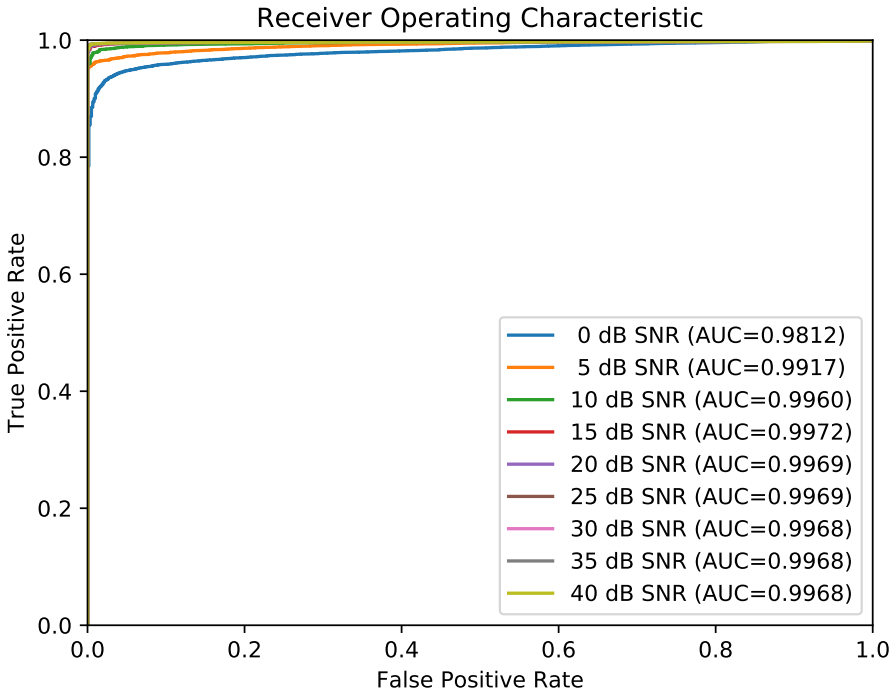


(c) Sample generated signal

Figure 6.2: The real (blue) and imaginary (green) components of an example (a) ideal signal, (b) measured signal, and (c) generated signal.



(a) ROC without GAN-augmented training



(b) ROC with GAN-augmented training

Figure 6.3: ROC curves to measure the ability to discriminate real transmissions from the target device (positive class) from adversarial transmissions generated by a GAN (negative class). (a) shows the results when GAN transmissions are not included in the training set. (b) shows the results when GAN transmissions are included in the training set.

Chapter 7: Toward Receiver-Agnostic RF Fingerprint Verification

7.1 Introduction

So far, the techniques presented have all assumed that the receiver used for training and evaluation of the RF fingerprint models is held constant. In a practical application, a factory producing devices may record transmissions from each device and train a fingerprint verification neural network to confirm the identity of that particular device. However, when the model is deployed to another receiver in a consumer's network, a different RF front-end will be used to verify the fingerprint of each transmission. As we show in this chapter, significant performance degradation may occur if techniques to mitigate the differences between receivers are not used. To counter this performance degradation, currently, every transmitter would have to be recorded by each receiver for optimal performance. This, of course, does not scale to IoT applications with billions of devices in use, so practical approaches for mitigating the degradation are needed.

Despite the recent influx of new techniques to perform RF fingerprinting, work to address receiver variability has been limited. Rehman *et al.* [112, 113] and Ramsey *et al.* [114] showed that RF fingerprinting algorithms based on power spectral density and amplitude/phase/frequency statistics, respectively, are practical to per-

form using lower-cost USRPs as receivers instead of high-end lab equipment, with a non-trivial performance penalty. However, their work assumed that a fingerprint model was trained for each receiver, and did not evaluate the case where a trained model was transferred for use with a different receiver.

Rehman *et al.* [115] demonstrated unacceptable performance degradation when a fingerprint model is used with another receiver. Their work shows experimental results when training a power spectral density-based classifier using a single low-end or high-end receiver, then evaluating the trained classifier with six different low-end receivers. They show that during inference, the classifier tends to nearly exclusively assign transmissions to a single output class, regardless of the true class of the input signal. Peng *et al.* [29] demonstrated significant, though less severe, performance degradation for a neural network-based fingerprinting system, measuring more than twice the error rate when inference is performed on another receiver. However, the data collection with the second receiver was performed 18 months after the training data collection, so the degradation due exclusively to the change in receiver are unclear. None of the above approaches have proposed methods to reduce this performance gap.

In this chapter, we demonstrate a technique to reduce the performance gap between the receiver used for training and others. First, we confirm that the performance of a neural network-based RF fingerprint verification system is suboptimal when the receiver used for training and inference are different. We then demonstrate that this performance gap can be reduced either by training with several physical or simulated receivers, or by learning a simple ResNet [9] model per receiver to

transform signals prior to classification.

7.2 Methods and Results

This section will show experimentally that performing training and inference on different receivers causes significant degradation in classifier performance. Then, we will outline an experiment to assess the impact of training using data recorded on multiple receivers. Next, we propose a technique for learning a single transformation function per receiver which can be used to adapt the received signals for fingerprinting before presentation to the network, and compare this to a simple LTI model. Then, we combine the multi-receiver training regime with the transformation method to permit training with several virtual receivers in the presence of only a single physical receiver. Finally, we will show that even if the receiver used for training models and transformations is no longer available, that we can replace it with another receiver using the described transformation method to maintain high accuracy. All pre-processing and training of fingerprint verification networks follows the techniques outlined in Chapter 5.

7.2.1 Data Collection

First, each ZigBee device is networked with another identical device, each inside of a separate sealed RF enclosure, with signals received by an antenna inside of each enclosure. The signals are combined using a power splitter, and the resulting time-series signal is downconverted to complex baseband and digitized at 16 MHz

using each of the ten available receivers. This process is repeated for each transmitter/receiver combination using 25 identical Digi XBP24CZ7SITB003 ZigBee transmitters and ten receivers: one Rohde & Schwarz FSW67 signal and spectrum analyzer, one Deepwave Digital artificial intelligence radio transceiver (AIR-T), and eight Ettus Research X300 USRPs.

7.2.2 Effect of Changing the Receiver

To fully characterize the performance degradation introduced by changing receivers between training and inference, two measurements are required. First, we measure the performance when training and inference are performed on the same receiver for all transmitter/receiver combinations. Next, we use only the networks trained on data recorded using a single receiver and perform inference using data from each of the other receivers individually. For comparison, we measure the distance of each classifier at each SNR from the ideal classifier, or the difference between the AUC and the ideal AUC of 1.0. Denoting the mean AUC of the ROC across all transmitters for classifiers trained on receiver a and evaluated on receiver b at a particular SNR as $\mathrm{AUC}_{a,b}^{\mathrm{SNR}}$, we say that the same-receiver case performs better by a factor of $G_{a,b}^{\mathrm{SNR}}$, where

$$G_{a,b}^{\mathrm{SNR}} = 1 - \frac{1 - \mathrm{AUC}_{b,b}^{\mathrm{SNR}}}{1 - \mathrm{AUC}_{a,b}^{\mathrm{SNR}}}. \tag{7.1}$$

Then, the average performance increase by using the same receiver for training and inference may be computed by averaging $G_{a,b}^{\mathrm{SNR}}$ over all evaluated receiver

combinations. For this experiment, we use FSW67 as receiver a, and each of the nine remaining receivers as receiver b, then average the results of each of the nine $G_{a,b}^{\text{SNR}}$ values at each SNR across receivers. Note that for all experiments, only 20 ZigBee devices are used for evaluation, as five devices are withheld to learn the transformation functions. The results are shown in Table 7.1.

SNR (dB)	**Mean ROC AUC**		**Mean Improvement(%)**
	Diff. Receiver	**Same Receiver**	
0	0.8558	0.8708	**10.35**
5	0.9311	0.9480	**24.37**
10	0.9582	0.9761	**42.32**
15	0.9639	0.9862	**60.24**
20	0.9667	0.9892	**65.71**
25	0.9656	0.9900	**69.05**
30	0.9651	0.9907	**71.45**
35	0.9667	0.9899	**67.57**
40	0.9644	0.9903	**70.83**

Table 7.1: The AUC ROC at each SNR averaged over nine receivers, considering both the case where the same receiver and different receivers were used for training and inference. Additionally, the mean improvement when using the same receiver is calculated using Equation 7.1.

From the results, it is clear that significant classifier performance degradation occurs when training and inference are performed using different receivers. The same-receiver setup achieves 70.83% better performance than the different-receiver setup in the 40 dB SNR case, and 10.35% better performance in the 0 dB SNR case. There is a strong upward trend, with the performance improvement typically being stronger at higher SNRs. This is partially due to the use of relative metrics, with the performance at lower SNRs being lower to begin with. Still, there is a clear, significant performance penalty associated with changing receivers.

7.2.3 Multi-Receiver Training

One obvious way to attempt to overcome this issue is by using recordings from a diverse set of receivers to train the fingerprint models. Unfortunately, receivers are typically expensive, and coordinating data collection from a large number of receivers is a complex task, so the application of this technique will typically be limited to a small number of receivers.

In this experiment, we retrain each of the fingerprint models for each of the 20 devices using $\{2, 3, \ldots, 9\}$ receivers. The total size of the training and validation sets are held constant, and the data is taken approximately evenly from each receiver used for training. After training, we measure the performance of each model on signals recorded from all 20 devices on the remaining receivers. Finally, we compute the improvement over the single receiver case using a modified metric,

$$G_{A,b}^{\mathrm{SNR}} = 1 - \frac{1 - \mathrm{AUC}_{A,b}^{\mathrm{SNR}}}{1 - \mathrm{AUC}_{A_1,b}^{\mathrm{SNR}}}, \tag{7.2}$$

where A is now a set of receivers used for training, b is each individual receiver used for inference, and A_1 denotes the first element in A, or the FSW67 receiver used for single-receiver training. This metric is computed for each receiver $b, b \notin A$, and then the mean is taken. The results are shown in Table 7.2, and indicate that training with multiple receivers significantly improves the generalizability of the fingerprint models to new receivers.

# of Receivers	Metric	SNR (dB)								
		0	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**
1	Mean ROC AUC	0.8558	0.9311	0.9582	0.9639	0.9667	0.9656	0.9651	0.9667	0.9644
	Improvement (%)	—	—	—	—	—	—	—	—	—
2	Mean ROC AUC	0.8766	0.9468	0.9721	0.9795	0.9839	0.9835	0.9837	0.9848	0.9832
	Improvement (%)	**14.43**	**22.77**	**33.17**	**43.34**	**51.67**	**51.91**	**53.44**	**54.32**	**52.88**
3	Mean ROC AUC	0.8579	0.9350	0.9667	0.9783	0.9836	0.9844	0.9845	0.9854	0.9836
	Improvement (%)	**1.48**	**5.65**	**20.39**	**39.98**	**50.82**	**54.66**	**55.72**	**56.23**	**54.05**
4	Mean ROC AUC	0.8646	0.9405	0.9717	0.9817	0.9867	0.9875	0.9877	0.9885	0.9870
	Improvement (%)	**6.08**	**13.58**	**32.15**	**49.46**	**60.22**	**63.52**	**64.73**	**65.54**	**63.52**
5	Mean ROC AUC	0.8655	0.9423	0.9717	0.9816	0.9865	0.9877	0.9876	0.9884	0.9872
	Improvement (%)	**6.75**	**16.26**	**32.15**	**49.21**	**59.60**	**64.32**	**64.58**	**65.13**	**63.97**
6	Mean ROC AUC	0.8578	0.9342	0.9676	0.9795	0.9856	0.9864	0.9861	0.9866	0.9856
	Improvement (%)	**1.40**	**4.44**	**22.38**	**43.15**	**56.77**	**60.39**	**60.32**	**59.73**	**59.45**
7	Mean ROC AUC	0.8526	0.9334	0.9702	0.9826	0.9871	0.9875	0.9880	0.9888	0.9873
	Improvement (%)	**-2.21**	**3.33**	**28.71**	**51.77**	**61.23**	**63.79**	**65.66**	**66.43**	**64.39**
8	Mean ROC AUC	0.8731	0.9447	0.9724	0.9827	0.9860	0.9866	0.9867	0.9874	0.9868
	Improvement (%)	**12.01**	**19.70**	**33.93**	**52.10**	**57.99**	**60.94**	**61.96**	**62.25**	**62.79**
9	Mean ROC AUC	0.8678	0.9344	0.9684	0.9826	0.9864	0.9883	0.9886	0.9884	0.9878
	Improvement (%)	**8.32**	**4.74**	**24.39**	**51.90**	**59.14**	**65.87**	**67.28**	**65.21**	**65.65**

Table 7.2: The mean ROC AUC across receivers used for inference when multiple receivers are used for training. The mean improvement relative to the single receiver case, calculated using Equation 7.2 at each receiver, is shown for each case.

In the best case, at 40 dB SNR, 65.65% improvement is noted by training on nine receivers. At lower SNRs, the results are much less consistant, with a best case improvement at 0 dB SNR of 14.43% measured using only two receivers. Comparing the mean improvement numbers in Tables 7.1 and 7.2, we see that at high SNRs, training with nine receivers provides only slightly less improvement over the different receiver case than training and testing on the same receiver. This demonstrates that training on multiple receivers improves the generalization at high SNRs. However, at lower SNRs, the benefits achieved with multi-receiver training are inconsistent and often fall short of the improvements achieved when training on the same receiver.

7.2.4 Learning Neural Network Receiver Transformations

While the performance of the multi-receiver training protocol is promising, it requires dedicating a large number of potentially costly receivers strictly to training, and also involves additional complexity in controlling and gathering data from all of the receivers simultaneously. As an alternative, we seek solutions in which the fingerprint models can be trained using only a single receiver, with a learned transformation on the inference receiver to adjust the signals it receives in an attempt to improve classification performance. In this model, signals would be recorded on the inference receiver, pre-preprocessed as before, transformed using the learned function, then presented to the fingerprint models trained using only a single receiver.

For this technique, we model our transformation as a ResNet [9]. In this manner, the transformed signal is simply the input signal plus some additive term

which is a function of the input signal, or

$$x_{\text{out}} = x_{\text{in}} + f(x_{\text{in}}). \tag{7.3}$$

where x_{out} is the transformed output signal, x_{in} is the pre-processed received signal, and $f(.)$ is a function learned by the neural network.

The ResNet model is intuitive for this application, since the transformed output signal can be expected to be very close to the input signal given that distortions in each receiver should be relatively small. The network architecture is shown in Table 7.3. Note that the normalization stage also includes RMS scaling and mean/variance scaling. Truncation of the leading samples is used so that signals are compatible with the lengths used in the verification networks, while the longer input signals are useful to avoid zero-padding the convolutions. Finally, note that the input signal is not an error signal, but the pre-processed signal prior to subtracting the estimated ideal signal.

To train the transformation layers, we propose the scheme shown in Figure 7.1. In this case, we reserve a set of five transmitters to be used solely for the transformation learning process. The fingerprint models are trained for these five devices using the techniques described in Chapter 5 using a single receiver, in this case FSW67. Then, signals from all five devices are passed through each of the five networks, and for each signal, the output distribution over all five networks is normalized to sum to one and saved. We denote this output distribution as $p = [p_1, p_2, \ldots, p_5]$. Next, signals recorded on the target receiver are passed through

Layer	Configuration	Activation	# Param.
Input	$18,432 \times 2$	—	—
Convolution 1D	64×32	ELU	$4,160$
Gaussian Noise	$\sigma = 0.15$	—	—
Convolution 1D	32×25	ELU	$51,232$
Gaussian Noise	$\sigma = 0.15$	—	—
Convolution 1D	4×19	ELU	$2,436$
Gaussian Noise	$\sigma = 0.15$	—	—
Convolution 1D	2×16	None	130
Add	Skip connection to input	—	—
Normalization	Produce error signal	—	—
Truncation	Discard leading samples	—	—
Output	$17,408 \times 2$	—	—

Table 7.3: The architecture of the transformation network. Note that the estimated ideal signal is also needed as input to the network to compute the error signal.

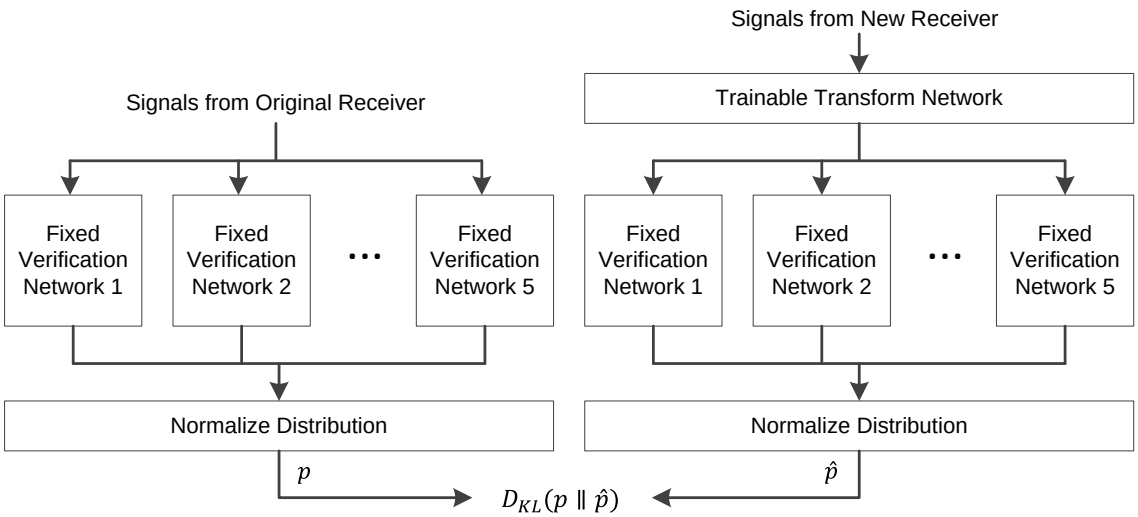


Figure 7.1: A diagram illustrating the procedure for learning a transformation function to map signals recorded on a new receiver to signals recorded on the training receiver. A set of five transmitters are used for calibration between the two receivers.

the trainable transformation network, then each of the five fixed fingerprint models, and finally normalized. We denote this distribution as $\hat{p} = [\hat{p_1}, \hat{p_2}, \ldots, \hat{p_5}]$.

If the transformation is perfect, we would expect the aggregate distributions output from each network to be quite similar for signals from the same transmitting device and at the same SNR. To compare distributions, we used the Kullback-Leibler divergence as the loss function, which we backpropagate back through the fixed transformation networks and into the trainable transformation layer to enable learning. The Kullback-Leibler divergence is defined as

$$D_{KL}(p \parallel \hat{p}) = -\sum_i p_i \log \frac{\hat{p}_i}{p_i}, \tag{7.4}$$

and is used as a measure of similarity between probability distributions. By minimizing this measure, we ensure that the distribution of outputs from the original receiver and the transformed new receiver are similar.

Note that during data collection, each receiver recorded different signals from each device. As such, we cannot simply assign the target probability distribution from a transmission on one receiver to that of the corresponding transmission on the original receiver. Instead, we randomly select a transmission from the same device and at the same SNR from the original receiver, and use that distribution as the target probability distribution for the training and validation datasets.

During the training process, we use 2,400 transmissions for training and 300 transmissions for validation from each of the five transmitter devices recorded on both the original and new receiver. Training proceeds for three epochs, and the

model with the highest ROC AUC on the validation dataset is saved as the transformation model for that receiver. ℓ^2 activity regularization with a weight of 10^{-5} is included on the final convolution layer to encourage adjustment to the input signal to be small.

To evaluate the performance of the learned transformation function, we consider a similar metric to Equations 7.1 and 7.2. However, based on the assumption that the optimal transformation would yield performance equivalent to models that are trained and evaluated on the same receiver, in this case, we measure the distance from the same-receiver classifier, instead of the ideal classifier. The resulting performance metric is given by

$$G_{a,b'}^{\mathrm{SNR}} = 1 - \frac{\mathrm{AUC}_{a,a}^{\mathrm{SNR}} - \mathrm{AUC}_{a,b'}^{\mathrm{SNR}}}{\mathrm{AUC}_{a,a}^{\mathrm{SNR}} - \mathrm{AUC}_{a,b}^{\mathrm{SNR}}}, \tag{7.5}$$

where b' indicates receiver b combined with the learned transformation for that receiver.

The average results for each receiver across all 20 fingerprint models at each SNR are shown in Table 7.4. From the results, it is clear that the transformation functions significantly improve classification performance across receivers. We measure a mean improvement of 50.11% in the 40 dB SNR case, and a mean improvement of 32.09% in the 0 dB SNR case, indicating that 50.11% and 32.09%, respectively, of the difference in performance between the same-receiver and different-receiver cases can be resolved by the proposed transformation process.

Receiver	Measurement	SNR (dB)								
		0	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**
X300-1	Without Transform	0.8637	0.9391	0.9696	0.9777	0.9818	0.9811	0.9826	0.9818	0.9812
	With Transform	0.8637	0.9454	0.9738	0.9811	0.9847	0.9863	0.9865	0.9871	0.9869
	Improvement (%)	**-2.40**	**100.41**	**60.91**	**38.63**	**40.22**	**56.40**	**46.60**	**63.67**	**58.94**
X300-2	Without Transform	0.8549	0.9368	0.9645	0.9725	0.9761	0.9768	0.9750	0.9776	0.9754
	With Transform	0.8626	0.9429	0.9737	0.9812	0.9844	0.9840	0.9847	0.9861	0.9851
	Improvement (%)	**72.81**	**70.71**	**76.93**	**61.49**	**64.36**	**52.88**	**60.95**	**67.56**	**62.65**
X300-3	Without Transform	0.8595	0.9326	0.9614	0.9672	0.9705	0.9707	0.9700	0.9719	0.9699
	With Transform	0.8597	0.9416	0.9713	0.9786	0.9818	0.9819	0.9818	0.9828	0.9822
	Improvement (%)	**3.48**	**70.30**	**66.03**	**58.31**	**60.93**	**57.03**	**56.31**	**59.37**	**58.67**
AIR-T	Without Transform	0.8550	0.9361	0.9592	0.9663	0.9690	0.9673	0.9670	0.9675	0.9657
	With Transform	0.8589	0.9375	0.9649	0.9689	0.9725	0.9702	0.9708	0.9725	0.9724
	Improvement (%)	**36.82**	**14.92**	**33.26**	**12.57**	**17.52**	**12.68**	**16.02**	**22.27**	**26.89**
X300-4	Without Transform	0.8579	0.9336	0.9591	0.9645	0.9673	0.9645	0.9654	0.9658	0.9651
	With Transform	0.8584	0.9390	0.9674	0.9748	0.9761	0.9764	0.9764	0.9786	0.9760
	Improvement (%)	**5.39**	**46.46**	**47.96**	**46.30**	**40.53**	**45.71**	**43.12**	**52.38**	**42.09**
X300-5	Without Transform	0.8536	0.9278	0.9563	0.9601	0.9617	0.9623	0.9613	0.9626	0.9604
	With Transform	0.8596	0.9369	0.9654	0.9703	0.9704	0.9692	0.9702	0.9724	0.9712
	Improvement (%)	**50.72**	**52.11**	**45.16**	**38.41**	**32.09**	**24.63**	**30.28**	**35.49**	**35.65**
X300-6	Without Transform	0.8551	0.9261	0.9533	0.9577	0.9592	0.9585	0.9576	0.9604	0.9579
	With Transform	0.8575	0.9411	0.9678	0.9726	0.9745	0.9731	0.9747	0.9737	0.9741
	Improvement (%)	**22.99**	**77.88**	**62.89**	**51.44**	**51.25**	**45.73**	**51.26**	**44.53**	**49.13**
X300-7	Without Transform	0.8511	0.9257	0.9514	0.9547	0.9582	0.9560	0.9545	0.9569	0.9524
	With Transform	0.8595	0.9386	0.9638	0.9713	0.9731	0.9722	0.9728	0.9734	0.9725
	Improvement (%)	**58.49**	**65.87**	**49.56**	**51.90**	**48.58**	**47.25**	**50.26**	**49.50**	**52.46**
X300-8	Without Transform	0.8514	0.9223	0.9493	0.9542	0.9563	0.9533	0.9525	0.9553	0.9518
	With Transform	0.8571	0.9380	0.9665	0.9730	0.9750	0.9746	0.9747	0.9761	0.9769
	Improvement (%)	**40.46**	**68.25**	**63.36**	**57.92**	**57.20**	**57.34**	**57.99**	**59.58**	**64.49**
All	**Mean Improvement (%)**	**32.09**	**62.99**	**56.23**	**46.33**	**45.85**	**44.41**	**45.86**	**50.49**	**50.11**

Table 7.4: The mean ROC AUC for each receiver at each SNR with and without the learned transformation function, and the improvement achieved by the inclusion of the transformation, calculated using Equation 7.5.

As a final comparison of the proposed method, we model the transformation from one receiver to another as a LTI system such that

$$x_a[n] = x_b[n] * h[n], \tag{7.6}$$

where $x_a[n]$ and $x_b[n]$ denote that signals, indexed by n, recorded on receivers a and b, respectively, $h[n]$ denotes the impulse response, and $*$ denotes convolution.

We then fit the impulse response $h[n]$ of the transformation using the following technique. First, we split the device output signal using a power splitter so that the same signal could be recorded on FSW67 and the worst-performing remaining receiver, in this case, X300-8. We pre-process signals from both receivers using the same process as before, but excluding the error signal generation and mean/variance scaling, then stack the signals from FSW67 into vector y, and form matrix $X = [x_0\, x_1\, \ldots\, x_{N-1}]$, where x_i is produced by stacking the corresponding signals recorded on X300-8 and shifting by i samples. Finally, we select the select the impulse response as

$$h = \text{argmin}_h \|y - Xh\|, \tag{7.7}$$

where $N = 200$ was found to produce the best results for this experiment.

The results comparing the linear transformation to the proposed transformation, as well as the same-receiver case and different-receiver case with no transformations, are shown in Figure 7.2 for the particular case of 40 dB SNR. From these results, it is clear that the proposed transformation is superior to both the

case with the linear transformation and the case with no transformation. Note that the difference is especially pronounced at relatively low values of the FPR, where a typical RF fingerprint classification system is likely to operate. At every SNR, the proposed transformation demonstrates superior performance, with the linear transformation achieving 6.48% improvement at 0 dB SNR and 33.96% improvement at 40 dB SNR, with a maximum of 37.20% improvement at 25 dB SNR, calculated using Equation 7.5. At the same SNRs, the proposed transformation achieves improvements of 40.46%, 64.49%, and 57.34%, respectively.

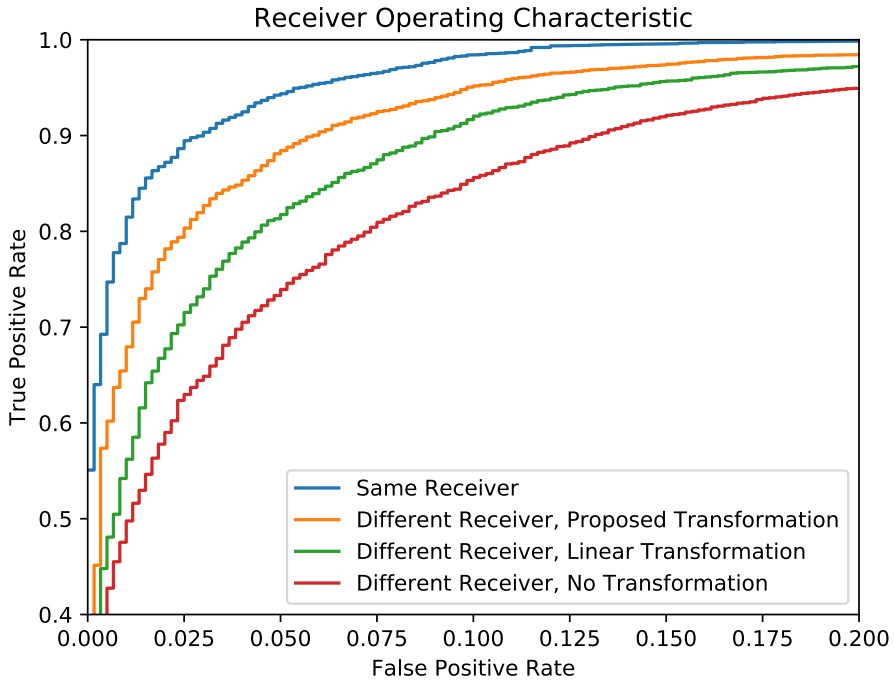


Figure 7.2: The average ROC curve at 40 dB SNR computed across all 20 devices on X300-8 for four cases, including training and testing on the same receiver and different receivers with each of the following configurations: the proposed transformation, the linear transformation, and no transformation.

7.2.5 Multi-Receiver Training with Virtual Receivers

The results of Section 7.2.3 illustrate the clear benefit to using multiple receivers during the training process. Unfortunately, this benefit comes at increased financial cost, since additional receivers must be purchased and maintained. Additionally, the data collection process becomes substantially more complex, since numerous receivers must be controlled, and the data from all receivers must be aggregated to build a training dataset. These additional complications motivate the use of *virtual receivers* instead of physical ones. Using the transformations outlined in Section 7.2.4, we propose collecting data on a single physical receiver, transforming the received signal to representations more like several other receivers, and combining these transformed signals into a single dataset to train a fingerprint model. This solution is depicted in Figure 7.3.

In order to train the transformation functions, temporary access to other physical receivers is necessary. However, once these transformations are learned, only a single physical receiver is required. This provides a clear benefit, since the remaining receivers may be rented temporarily and do not need to be maintained. Additionally, this setup simulates the virtual receivers in software, so that only a single physical receiver is used to collect data, permitting a simple data collection process. Finally, this technique places the computational burden of using transformations on the training side, removing the need for transformations to be used for inference.

Following the steps of Section 7.2.4, we train three transformation functions

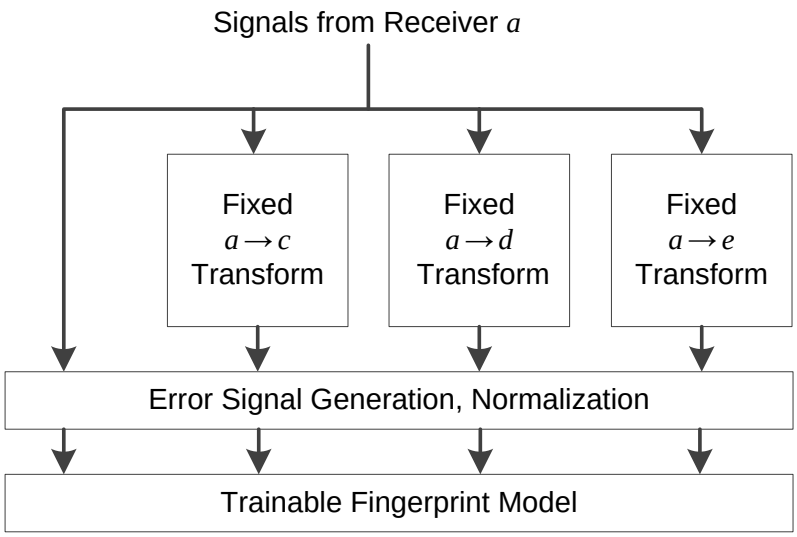


Figure 7.3: The proposed system for training the fingerprint models using a single physical receiver, a, and three virtual receivers, c, d, and e.

to map the FSW67 receiver to the AIR-T, X300-1, and X300-2 receivers. Next, the weights of the fingerprint model are initialized to the weights trained for a verification network for one of the five withheld transmitters used for calibration. Note that these verification networks on the set of five calibration devices are already trained, since they are required to learn the transformation functions for each receiver. The neural network structure in Figure 7.3 is then used to train the fingerprint models, with training disabled on the transformation weights.

Training proceeds as in Chapter 5, except 10% of the training dataset from FSW67 is randomly replaced with data selected uniformly from the virtual receivers. All other training hyperparameters are held constant.

To evaluate this technique, data from the remaining receivers, namely X300-3 through X300-8, is used to calculate the ROC AUC for each of the 20 fingerprint models for each receiver. These are the exact same datasets that were used to calculate the performance in Section 7.2.2 where training and inference were performed

on different receivers. The results are shown in Table 7.5, and indicate significant performance improvement over the case using only a single physical receiver, particularly at lower SNRs. At 0 dB SNR, we observe a 29.08% mean improvement, and at 40 dB SNR, we observe a 10.24% mean improvement, over the single training receiver case.

Interestingly, these results indicate a greater benefit to using multiple virtual receivers than multiple physical receivers in the 0 dB and 5 dB SNR cases. Additionally, comparing mean ROC AUC numbers, one notices that this method also provides higher performance in the 0 dB and 5 dB SNR cases than the learned transformation functions in nearly all cases. At higher SNRs, however, additional physical receivers and learned transformation functions provide greater performance. We hypothesize that this occurs at the lower SNRs because the noise makes it more difficult for the network to learn patterns in the data, which is assisted by showing the network the same signal as perceived from several different receivers, both physical and virtual.

Receiver	Measurement	SNR (dB)								
		0	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**
X300-3	Mean ROC AUC	0.9025	0.9525	0.9668	0.9700	0.9732	0.9732	0.9725	0.9754	0.9725
	Improvement (%)	**30.62**	**29.54**	**14.19**	**8.53**	**8.93**	**8.49**	**8.37**	**12.40**	**8.67**
X300-4	Mean ROC AUC	0.8995	0.9483	0.9624	0.9676	0.9695	0.9691	0.9690	0.9696	0.9683
	Improvement (%)	**29.24**	**22.22**	**8.00**	**8.73**	**6.51**	**12.87**	**10.35**	**11.21**	**9.07**
X300-5	Mean ROC AUC	0.8967	0.9442	0.9594	0.9635	0.9658	0.9655	0.9640	0.9663	0.9648
	Improvement (%)	**29.44**	**22.77**	**7.07**	**8.56**	**10.84**	**8.62**	**7.10**	**10.00**	**11.34**
X300-6	Mean ROC AUC	0.8960	0.9427	0.9570	0.9607	0.9640	0.9635	0.9614	0.9650	0.9623
	Improvement (%)	**28.23**	**22.42**	**7.90**	**7.11**	**11.70**	**12.09**	**9.00**	**11.54**	**10.45**
X300-7	Mean ROC AUC	0.8944	0.9403	0.9546	0.9577	0.9606	0.9600	0.9588	0.9608	0.9572
	Improvement (%)	**29.10**	**19.67**	**6.60**	**6.64**	**5.73**	**9.21**	**9.44**	**8.91**	**10.05**
X300-8	Mean ROC AUC	0.8927	0.9376	0.9526	0.9586	0.9598	0.9588	0.9565	0.9597	0.9573
	Improvement (%)	**27.82**	**19.59**	**6.43**	**9.62**	**8.09**	**11.68**	**8.57**	**9.74**	**11.87**
All	**Mean Improvement (%)**	**29.08**	**22.70**	**8.37**	**8.20**	**8.63**	**10.49**	**8.81**	**10.63**	**10.24**

Table 7.5: The mean ROC AUC for each remaining receiver at each SNR when training is performed using FSW67 and virtual receivers trained using AIR-T, X300-1, and X300-2. The improvement over the single receiver case, calculated using Equation 7.2, is also presented.

7.2.6 Training without the Original Receiver

The transformation method in Section 7.2.4 makes the implicit assumption that all fingerprint models are trained using the same receiver, and all devices used for inference learn a transformation function to that original receiver. One might imagine a scenario where a factory producing transceivers records transmissions from each device to train a fingerprint model for that device, then the new transceiver records the five transmitters used for calibration, and a transformation function to the training receiver is learned. The problem with this approach is that if the original receiver later becomes unavailable, perhaps lost or damaged, future fingerprint models will have to be trained on a different receiver, invalidating the transforms on existing devices.

One possible solution to this problem is to use a transformation function to map signals recorded on the new training receiver to the original training receiver, and to then treat the newly transformed receiver as if it were the original. At inference time, the devices will behave as if this switch never occurred, using the original transformation function to the now broken receiver, and the fingerprint models trained using the transformed new receiver.

This approach is shown in Figure 7.4. Note that the $c \rightarrow a$ transformation can be learned for new receiver c even after receiver a is no longer available, as long as signals recorded using receiver a from each of the five calibration devices have been saved and those five devices are still available to be recorded by receiver c.

We tested this approach experimentally, using AIR-T as receiver c and learn-

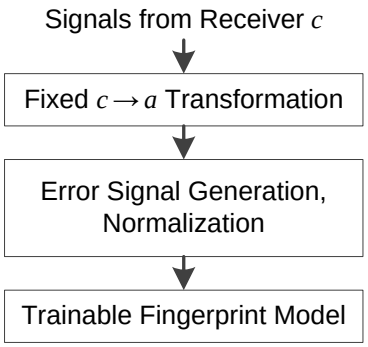


Figure 7.4: The proposed method for training new fingerprint models after the original receiver, a, is no longer available and is replaced by receiver c.

ing the $c \rightarrow a$ transformation as described in Section 7.2.4. Next, we retrained fingerprint models for each of the 20 transmitters using the procedure outlined in Chapter 5, except with the transformed AIR-T receiver. In an effort to train fingerprint models that are as similar as possible to the ones that would be produced using the original receiver, FSW67, we initialize the weights in each model prior to training to the weights from a fingerprint model developed using the original receiver and one of the five devices for calibration.

To evaluate this technique, we use the same datasets as for the experiments in Section 7.2.4, transform the signals for each X300 receiver using the transformation to FSW67, and perform inference using the newly trained fingerprint models. As a comparison, we measure the improvement over the case where training is performed using AIR-T and inference is performed using each particular receiver, which means that the alternative is simply to disable the transformation to FSW67 on each receiver and use the models trained on AIR-T. We can measure the improvement using a modified version of Equation 7.5,

$$G^{\text{SNR}}_{c',b'} = 1 - \frac{\text{AUC}^{\text{SNR}}_{c,c} - \text{AUC}^{\text{SNR}}_{c',b'}}{\text{AUC}^{\text{SNR}}_{c,c} - \text{AUC}^{\text{SNR}}_{c,b}}, \tag{7.8}$$

where training receiver a is replaced with receiver c, or AIR-T, b' indicates receiver b transformed to receiver a, or FSW67, and c' indicates receiver c transformed to receiver a.

The results of the experiment are shown in Table 7.6. Note that in nearly all of the 0 dB and 5 dB SNR cases, the performance of the new technique exceeded the performance of the case where training and inference were performed using the same receiver without any transformations. In this case, the interpretation of Equation 7.8 breaks down, so these entries are simply denoted with an asterisk. We hypothesize that a minor denoising effect is performed by the transformation network, permitting higher performance in the lower SNR cases than is achieved when training without a transformation in place.

At the remaining SNRs, performance is impressive, achieving a maximum mean improvement of 57.06% at 10 dB SNR and a mean improvement of 41.92% in the 40 dB SNR case. Compared to the results in Table 7.4, which indicate performance when models are learned on the original, untransformed receiver and where we note 56.23% improvement at 10 dB SNR and 50.11% improvement at 40 dB SNR, we note that these results are quite promising. While the model performance is impacted by replacing the original training receiver with another, transformed receiver, the resulting performance is a significant improvement over the case where training and inference are performed on different receivers without transformation.

Receiver	Measurement	SNR (dB)								
		0	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**
X300-1	Mean ROC AUC	0.8804	0.9431	0.9677	0.9742	0.9745	0.9763	0.9751	0.9764	0.9755
	Improvement (%)	*	**90.77**	**39.64**	**18.22**	**6.15**	**18.60**	**10.58**	**16.47**	**19.31**
X300-2	Mean ROC AUC	0.8788	0.9436	0.9687	0.9774	0.9770	0.9787	0.9775	0.9800	0.9781
	Improvement (%)	*	*	**26.90**	**33.03**	**4.98**	**26.01**	**16.55**	**31.82**	**26.51**
X300-3	Mean ROC AUC	0.8767	0.9446	0.9695	0.9776	0.9783	0.9797	0.9791	0.9809	0.9802
	Improvement (%)	*	*	**43.83**	**36.59**	**24.90**	**38.00**	**30.40**	**30.89**	**37.45**
X300-4	Mean ROC AUC	0.8807	0.9453	0.9693	0.9777	0.9778	0.9798	0.9782	0.9806	0.9799
	Improvement (%)	*	*	**46.67**	**46.63**	**31.85**	**48.48**	**31.77**	**45.42**	**43.67**
X300-5	Mean ROC AUC	0.8819	0.9465	0.9724	0.9801	0.9810	0.9818	0.9812	0.9824	0.9828
	Improvement (%)	*	*	**88.75**	**64.68**	**41.82**	**51.74**	**43.37**	**46.57**	**54.77**
X300-6	Mean ROC AUC	0.8788	0.9440	0.9703	0 9775	0.9784	0.9794	0.9783	0.9791	0.9789
	Improvement (%)	*	*	**72.06**	**54.99**	**38.09**	**48.48**	**37.96**	**43.53**	**45.75**
X300-7	Mean ROC AUC	0.8783	0.9460	0.9695	0.9766	0.9785	0.9778	0.9786	0.9787	0.9788
	Improvement (%)	*	*	**65.49**	**57.94**	**48.24**	**51.08**	**50.63**	**48.90**	**55.23**
X300-8	Mean ROC AUC	0.8807	0.9442	0.9699	0.9744	0.9757	0.9768	0.9759	0.9772	0.9772
	Improvement (%)	*	*	**73.12**	**45.10**	**42.67**	**50.70**	**46.52**	**47.92**	**52.65**
All	**Mean Improvement (%)**	*	*	**57.06**	**44.65**	**29.84**	**41.64**	**33.47**	**38.94**	**41.92**

Table 7.6: The mean ROC AUC for each remaining receiver at each SNR when training is performed using AIR-T and the AIR-T→FSW67 transformation, and evaluated using the transformation to FSW67 for each receiver. The improvement over the single receiver (AIR-T) case, calculated using Equation 7.8, is also presented. * indicates cases where the performance exceeded the same-receiver performance.

7.3 Conclusion

In this chapter, we demonstrated that RF fingerprint verification network performance is negatively impacted when the receiver used for training and testing are different. We proposed and evaluated several techniques for overcoming this shortcoming. The first technique simply trains the verification networks on signals from several receivers. This technique demonstrates significant performance improvement at high SNR, but inconsistent, yet mostly positive, improvement at low SNR. Next, we demonstrated a calibration procedure whereby a neural network transformation is learned for each receiver and used with verification networks trained on another receiver. This technique is able to reduce the performance gap introduced by testing on a new receiver by between 32.09% and 62.99% depending on SNR, and eliminates the scaling issue introduced by training a new model for every transmitter/receiver pair.

We then demonstrate additional experiments using the proposed transformation functions. First, we show that increased generalization over single-receiver training may be achieved by using the transformation functions to simulate additional receivers, even when only a single physical receiver is used for data collection. Then, we demonstrate that the transformation functions may be used during the training process to replace the original receiver with another in the event that the original receiver is no longer available.

Chapter 8: Conclusion

In this dissertation, we have developed several successful deep learning techniques for the RF fingerprinting problem and demonstrated them on a set of commercial ZigBee devices. The first part of our work develops a framework for which to derive the estimated error signal for a received transmission, then trains CNNs to classify the error signals by the predicted device. The second part builds upon the first and utilizes RNN components and an improved pre-processing sequence to improve the performance of the technique, particularly at lower SNRs, and also performs several experiments to measure the success of the technique against potential realistic scenarios. The third part develops a GAN framework for which to generate artificial signals, and shows that using a GAN to provide supplementary training examples for the classifiers provides additional protection against these adversarial examples. The final contribution of our work investigates the ability to transfer a RF fingerprint model from one receiver to another, and develops a calibration procedure whereby the front-end of the evaluation receiver can be mapped to the front-end of the training receiver using a neural network model, improving upon the performance of a fingerprint model that is blindly transferred without compensation.

A number of future directions for this work remain. First, an analysis of the

stability of an RF fingerprint model over time and through temperature and voltage fluctuations is important. While the techniques presented here work well, they are limited to transmitters that have been measured in a somewhat fixed environment and do not account for any time-varying effects that occur slowly relative to the capture length of each recording, such as temperature deviation in the environment. IoT devices are also typically battery-operated, and an investigation into any possible changes that develop as battery levels decrease are necessary before a practical implementation of such techniques. Tekbaş *et al.* performed experiments to show that a transient-based fingerprinting technique degrades significantly as a result of either battery voltage or temperature deviation, but also showed that including these deviations in the training set allows the algorithm to regain nearly all of the lost performance [116]. It would be useful to verify whether a similar performance degradation may be seen in neural network fingerprinting models, and determine if including such variability in the training set is sufficient to compensate for such effects.

Second, techniques aimed at reducing the time-complexity of neural network fingerprinting algorithms are needed. Near real-time inference using the proposed algorithms may be attainable using high-performance workstations with several GPUs, but resources are typically much more limited in the IoT domain. The continued improvement of deep learning software packages and the development of new hardware platforms geared towards neural network applications may also enhance performance.

Finally, additional techniques to evaluate what has been learned by a particular

neural network model are critical to their increased use within the security domain. Given the nature of security applications, explainable algorithms are strongly desired as they allow an understanding of the limitations and possible exploitations of a particular approach. Furthermore, such a development could permit algorithms whereby domain-specific attributes are intentionally encoded into the network, allowing for expert knowledge and artificial intelligence to cooperatively solve the RF fingerprinting problem.

Bibliography

[1] *IEEE Standard for Local and Metropolitan Area Networks-Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, IEEE Standards Association Std., Rev. 2011, Sep. 2011.

[2] *ZigBee Specification*, ZigBee Alliance, Inc. Std., Rev. 20, Sep. 2012.

[3] K. Merchant, S. Revay, G. Stantchev, and B. Nousain, "Deep learning for RF device fingerprinting in cognitive communication networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 160–167, Feb. 2018.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[5] K. Merchant and B. Nousain, "Enhanced RF fingerprinting for IoT devices with recurrent neural networks," in *IEEE Military Communications Conference (MILCOM)*, Norfolk, VA, Nov. 2019.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair *et al.*, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, 2014, pp. 2672–2680.

[7] K. Merchant and B. Nousain, "Securing IoT RF fingerprinting systems with generative adversarial networks," in *IEEE Military Communications Conference (MILCOM)*, Norfolk, VA, Nov. 2019.

[8] K. Merchant and B. Nousain, "Toward receiver-agnostic RF fingerprint verification," in *IEEE Global Communications Conference (GLOBECOM) Workshops*, Waikoloa, HI, Dec. 2019.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778.

[10] P. Tilghman. Radio frequency machine learning systems (RFMLS). Defense Advanced Research Projects Agency. [Online]. Available: https://www.darpa.mil/program/radio-frequency-machine-learning-systems

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, pp. 1097–1105.

[12] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 1725–1732.

[13] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 689–696.

[14] C. K. Dubendorfer, B. W. Ramsey, and M. A. Temple, "An RF-DNA verification process for zigbee networks," in *IEEE Military Communications Conference*, Orlando, FL, Oct./Nov. 2012.

[15] H. Patel, "Non-parametric feature generation for RF-fingerprinting on zigbee devices," in *Computational Intelligence for Security and Defense Applications (CISA), 2015 IEEE Symposium on*, Verona, NY, May 2015.

[16] T. J. Bihl, K. W. Bauer, and M. A. Temple, "Feature selection for RF fingerprinting with multiple discriminant analysis and using zigbee device emissions," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 8, pp. 1862–1874, Aug. 2016.

[17] C. Dubendorfer, B. Ramsey, and M. Temple, "Zigbee device verification for securing industrial control and building automation systems," in *International Conference on Critical Infrastructure Protection VII*, Washington, DC, Mar. 2013, pp. 47–61.

[18] Y. Yuan, Z. Huang, H. Wu, and X. Wang, "Specific emitter identification based on hilbert-huang transform-based time-frequency-energy distribution features," *IET Communications*, vol. 8, no. 13, pp. 2404–2412, Sep. 2014.

[19] A. M. Ali, E. Uzundurukan, and A. Kara, "Assessment of features and classifiers for bluetooth RF fingerprinting," *IEEE Access*, vol. 7, pp. 50 524–50 535, Apr. 2019.

[20] K. Yang, J. Kang, J. Jang, and H.-N. Lee, "Multimodal sparse representation-based classification scheme for RF fingerprinting," *IEEE Communications Letters*, vol. 23, no. 5, pp. 867–870, May 2019.

[21] C. Bertoncini, K. Rudd, B. Nousain, and M. Hinders, "Wavelet fingerprinting of radio-frequency identification (RFID) tags," *IEEE Trans. Ind. Electron.*, vol. 59, no. 12, pp. 4843–4850, Dec. 2012.

[22] F. Xie, H. Wen, Y. Li, S. Chen, L. Hu, Y. Chen *et al.*, "Optimized coherent integration-based radio frequency fingerprinting in internet of things," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3967–3977, Oct. 2018.

[23] X. Zhou, A. Hu, G. Li, L. Peng, Y. Xing, and J. Yu, "Design of a robust RF fingerprint generation and classification scheme for practical device identification," in *IEEE Conference on Communications and Network Security*, Jun. 2019.

[24] Y. Huang and H. Zheng, "Radio frequency fingerprinting based on the constellation errors," in *2012 18th Asia-Pacific Conference on Communications (APCC)*, Oct. 2012, pp. 900–905.

[25] A. Ali and G. Fischer, "Symbol based statistical RF fingerprinting for fake base station identification," in *2019 29th International Conference Radioelektronika (RADIOELEKTRONIKA)*, Apr. 2019, pp. 1–5.

[26] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proceedings of the 14th ACM international conference on Mobile computing and networking*, San Francisco, CA, Sep. 2008, pp. 116–127.

[27] Y. Pan, H. Peng, T. Li, and W. Wang, "TDMA device identification using continuity of carrier phase," *Journal of Physics: Conference Series*, vol. 1229, May 2019.

[28] A. Candore, O. Kocabas, and F. Koushanfar, "Robust stable radiometric fingerprinting for wireless devices," in *2009 IEEE International Workshop on Hardware-Oriented Security and Trust*, Jul. 2009, pp. 43–49.

[29] L. Peng, A. Hu, J. Zhang, Y. Jiang, J. Yu, and Y. Yan, "Design of a hybrid RF fingerprint extraction and device classification scheme," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 349–360, Feb. 2019.

[30] M.-W. Liu and J. F. Doherty, "Specific emitter identification using nonlinear device estimation," in *Sarnoff Symposium, 2008 IEEE*, Princeton, NJ, 2008, pp. 1–5.

[31] T. L. Carroll, "A nonlinear dynamics method for signal identification," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 17, no. 2, pp. 1–8, May 2007.

[32] T. L. Carroll, "Phase space method for identification of driven nonlinear systems," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 19, no. 3, pp. 1–8, August 2009.

[33] Y. Yuan, Z. Huang, F. Wang, and X. Wang, "Radio specific emitter identification based on nonlinear characteristics of signal," in *IEEE International Black Sea Conference on Communications and Networking, 2015*, Constanta, Romania, May 2015, pp. 77–81.

[34] S. Andrews, R. M. Gerdes, and M. Li, "Crowdsourced measurements for device fingerprinting," in *Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, May 2019, pp. 72–82.

[35] G. B. Willson, "Radar classification using a neural network," in *Proceedings of SPIE 1294, Applications of Artificial Neural Networks*, Aug. 1990, pp. 200–210.

[36] C.-S. Shieh and C.-T. Lin, "A vector neural network for emitter identification," *IEEE Trans. Antennas Propag.*, vol. 50, pp. 1120–1127, Aug. 2002.

[37] O. Üreten and N. Serinken, "Wireless security through RF fingerprinting," *Canadian Journal of Electrical and Computer Engineering*, vol. 32, no. 1, pp. 27–33, 2007.

[38] M. Köse, S. Taşcioğlu, and Z. Telatar, "RF fingerprinting of IoT devices based on transient energy spectrum," *IEEE Access*, vol. 7, pp. 18 715–18 726, Feb. 2019.

[39] H. Jafari, O. Omotere, D. Adesina, H.-H. Wu, and L. Qian, "IoT devices fingerprinting using deep learning," in *IEEE Military Communications Conference (MILCOM)*, Oct. 2018, pp. 1–9.

[40] Q. Wu, C. Feres, D. Kuzmenko, D. Zhi, Z. Yu, X. Liu *et al.*, "Deep learning based RF fingerprinting for device identification and wireless security," *Electronics Letters*, vol. 54, no. 24, pp. 1405–1407, Nov. 2018.

[41] S. Riyaz, K. Sankhe, S. Ioannidis, and K. Chowdhury, "Deep learning convolutional neural networks for radio identification," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 146–152, Sep. 2018.

[42] Y. Jiabao, A. Hu, L. Guyue, and L. Peng, "A multi-sampling convolutional neural network-based RF fingerprinting approach for low-power devices," in *Workshop on Internet of Things for Adversarial Environments*, Mar. 2019.

[43] K. Youssef, L. Bouchard, K. Haigh, J. Silovsky, B. Thapa, and C. V. Valk, "Machine learning approach to RF transmitter identification," *IEEE RFID J.*, vol. 2, no. 4, pp. 197–205, Dec. 2018.

[44] G. Baldini, R. Giuliani, and F. Dimc, "Physical layer authentication of internet of things wireless devices using convolutional neural networks and recurrence plots," *Internet Technology Letters*, Oct. 2018.

[45] Y. Pan, S. Yang, H. Peng, T. Li, and W. Wang, "Specific emitter identification based on deep residual networks," *IEEE Access*, vol. 7, pp. 54 425–54 434, May 2019.

[46] S. Gopalakrishnan, M. Cekic, and U. Madhow, "Robust wireless fingerprinting via complex-valued neural networks," *arXiv:1905.09388*, May 2019.

[47] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov *et al.*, "Going deeper with convolutions," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.

[48] S. Chen, S. Zheng, L. Yang, and X. Yang, "Deep learning for large-scale real-world ACARS and ADS-B radio signal classification," *arXiv:1904.09425*, Jun. 2019.

[49] L. J. Wong, "On the use of convolutional neural networks for specific emitter identification," Master's thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, 2018.

[50] S. S. Hanna and D. Cabric, "Deep learning based transmitter identification using power amplifier nonlinearity," in *2019 International Conference on Computing, Networking and Communications (ICNC)*, Feb. 2019, pp. 674–680.

[51] B. Chatterjee, D. Das, and S. Sen, "RF-PUF: IoT security enhancement through authentication of wireless nodes using in-situ machine learning," in *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, Apr. 2018, pp. 205–208.

[52] F. Restuccia, S. D'Oro, A. Al-Shawabka, M. Belgiovine, L. Angioloni, S. Ioannidis *et al.*, "DeepRadioID: Real-time channel-resilient optimization of deep learning-based radio fingerprinting algorithms," in *The Twentieth International Symposium on Mobile Ad Hoc Networking and Computing*, Catania, Italy, Jul. 2019.

[53] C. Morin, L. Cardoso, J. Hoydis, J. Gorce, and T. Vial, "Transmitter classification with supervised deep learning," *arXiv:1905.07923*, May 2019.

[54] N. Benvenuto, F. Piazza, and A. Uncini, "A neural network approach to data predistortion with memory in digital radio systems," in *IEEE International conference on communication, 1993*, Geneva, Switzerland, May 1993, pp. 232–236.

[55] F. Mkadem and S. Boumaiza, "Physically inspired neural network model for RF power amplifier behavioral modeling and digital predistortion," *IEEE Trans. Microw. Theory Techn.*, vol. 59, pp. 913–923, Jan. 2011.

[56] J. Wray and G. G. R. Green, "Calculation of the volterra kernels of non-linear dynamic systems using an artificial neural network," *Biological Cybernetics*, vol. 71, no. 3, pp. 187–195, Jul. 1994.

[57] H. Weng, X. Dong, X. Hu, D. G. Beetner, T. Hubing, and D. Wunsch, "Neural network detection and identification of electronic devices based on their unintended emissions," in *2005 International Symposium on Electromagnetic Compatibility*, Chicago, IL, Aug. 2005.

[58] X. Wang, L. Gao, S. Mao, and S. Pandey, "CSI-based fingerprinting for indoor localization: A deep learning approach," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 1, pp. 763–776, Jan. 2017.

[59] X. Wang, L. Gao, and S. Mao, "BiLoc: Bi-modal deep learning for indoor localization with commodity 5GHz WiFi," *IEEE Access*, vol. 5, pp. 4209–4220, Mar. 2017.

[60] X. Wang, X. Wang, and S. Mao, "ResLoc: Deep residual sharing learning for indoor localization with CSI tensors," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Oct. 2017, pp. 1–6.

[61] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, Dec. 2010.

[62] W. Zhang, K. Liu, W. Zhang, Y. Zhang, and J. Gu, "Deep neural networks for wireless localization in indoor and outdoor environments," *Neurocomputing*, vol. 194, pp. 279–287, Jun. 2016.

[63] H. Huang, J. Yang, H. Huang, Y. Song, and G. Gui, "Deep learning for super-resolution channel estimation and DOA estimation based massive MIMO system," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 9, pp. 8549–8560, Sep. 2018.

[64] G. J. Mendis, J. Wei, and A. Madanayake, "Deep learning-based automated modulation classification for cognitive radio," in *2016 IEEE International Conference on Communication Systems (ICCS)*, Dec. 2016, pp. 1–6.

[65] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 168–179, Feb. 2018.

[66] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Advances in Neural Information Processing Systems (NIPS) 28*, 2015, pp. 2017–2025.

[67] T. J. O'Shea, L. Pemula, D. Batra, and T. C. Clancy, "Radio transformer networks: Attention models for learning to synchronize in wireless systems," in *2016 50th Asilomar Conference on Signals, Systems and Computers*, Nov. 2016, pp. 662–666.

[68] M. Schmidt, D. Block, and U. Meier, "Wireless interference identification with convolutional neural networks," in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, Jul. 2017, pp. 180–185.

[69] N. Bitar, S. Muhammad, and H. H. Refai, "Wireless technology identification using deep convolutional neural networks," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Oct. 2017, pp. 1–6.

[70] S. Kokalj-Filipovic, R. Miller, and J. Morman, "Autoencoders for training compact deep learning RF classifiers for wireless protocols," *arXiv:1904.11874*, 2019.

[71] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI) 2015*, 2015, pp. 234–241.

[72] J. Akeret, C. Chang, A. Lucchi, and A. Réfrégier, "Radio frequency interference mitigation using deep convolutional neural networks," *Astronomy and Computing*, vol. 18, pp. 35–39, Jan. 2017.

[73] H. Rutagemwa, A. Ghasemi, and S. Liu, "Dynamic spectrum assignment for land mobile radio with deep recurrent neural networks," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, May 2018, pp. 1–6.

[74] M. Sadeghi and E. G. Larsson, "Adversarial attacks on deep-learning based radio signal classification," *IEEE Wireless Communications Letters*, vol. 8, no. 1, pp. 213–216, Feb. 2019.

[75] S. Kokalj-Filipovic, R. Miller, and J. Morman, "Targeted adversarial examples against RF deep classifiers," in *ACM Workshop on Wireless Security and Machine Learning (WiseML) 2019*, Apr. 2019, pp. 6–11.

[76] S. Kokalj-Filipovic, R. Miller, N. Chang, and C. L. Lau, "Mitigation of adversarial examples in RF deep classifiers utilizing autoencoder pre-training," *arXiv:1902.08034*, 2019.

[77] S. Kokalj-Filipovic and R. Miller, "Adversarial examples in RF deep learning: Detection of the attack and its physical robustness," *arXiv:1902.06044*, 2019.

[78] K. Davaslioglu and Y. E. Sagduyu, "Generative adversarial learning for spectrum sensing," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6.

[79] B. Tang, Y. Tu, Z. Zhang, and Y. Lin, "Digital signal modulation classification with data augmentation using generative adversarial nets in cognitive radio networks," *IEEE Access*, vol. 6, pp. 15 713–15 722, 2018.

[80] T. J. O'Shea, T. Roy, N. West, and B. C. Hilburn, "Physical layer communications system design over-the-air using adversarial networks," in *2018 26th European Signal Processing Conference (EUSIPCO)*, Sep. 2018, pp. 529–532.

[81] H. Ye, L. Liang, G. Y. Li, and B.-H. F. Juang, "Deep learning-based end-to-end wireless communication systems with conditional GAN as unknown channel," *arXiv:1903.02551*, 2019.

[82] F. Edalat, "Effect of power amplifier nonlinearity on system performance metric, bit-error-rate (BER)," Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 2003.

[83] C.-P. Liang, J.-H. Jong, W. E. Stark, and J. R. East, "Nonlinear amplifier effects in communications systems," *IEEE Trans. Microw. Theory Techn.*, vol. 47, no. 8, pp. 1461–1466, Aug. 1999.

[84] S. Benedetto, E. Biglieri, and R. Daffara, "Modeling and performance evaluation of nonlinear satellite links-a volterra series approach," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-15, no. 4, pp. 494–507, Jul. 1979.

[85] P. Singerl, "Complex baseband modeling and digital predistortion for wideband RF power amplifiers," Ph.D. dissertation, Graz University of Technology, Austria, Dec. 2006.

[86] L. Ding, G. T. Zhou, D. R. Morgan, Z. Ma, J. S. Kenney, J. Kim *et al.*, "A robust digital baseband predistorter constructed using memory polynomials," *IEEE Trans. Commun. Technol.*, vol. 52, no. 1, pp. 159–165, Jan. 2004.

[87] D. R. Morgan, Z. Ma, J. Kim, M. G. Zierdt, and J. Pastalan, "A generalized memory polynomial model for digital predistortion of RF power amplifiers," *IEEE Signal Process. Lett.*, vol. 54, no. 10, pp. 3852–3860, Oct. 2006.

[88] J. Hansen. (2013, Jul.) Vector modulation and frequency conversion fundamentals. Agilent Technologies. [Online]. Available: https://www.keysight.com/upload/cmc_upload/All/18July2013WebcastSlides.pdf

[89] R. Liu, Y. Li, H. Chen, and Z. Wang, "EVM estimation by analyzing transmitter imperfections mathematically and graphically," *Analog Integrated Circuits and Signal Processing*, vol. 48, no. 3, pp. 257–262, Sep. 2006.

[90] "Understanding data converters," Texas Instruments, Application Report SLAA013, 1995.

[91] P. Hendriks, "Specifying communications DACs," *IEEE Spectrum*, vol. 34, no. 7, pp. 58–69, Jul. 1997.

[92] M. Hasler and G. Kubin, "Mixed-domain system representation using volterra series," in *2008 IEEE International Symposium on Circuits and Systems*, May 2008, pp. 572–575.

[93] Digi XCTU. [Online]. Available: https://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu

[94] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," in *International Conference on Learning Representations*, 2016.

[95] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS-10)*, vol. 9, 2010, pp. 249–256.

[96] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[97] F. Chollet *et al.* Keras 2.1.6. [Online]. Available: https://github.com/fchollet/keras

[98] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro *et al.* (2015) TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[99] Y. Ren, L. Peng, W. Bai, and J. Yu, "A practical study of channel influence on radio frequency fingerprint features," in *2018 IEEE International Conference on Electronics and Communication Engineering (ICECE)*, Dec. 2018, pp. 1–7.

[100] T. W. Parks and J. H. McClellan, "Chebyshev approximation for nonrecursive digital filters with linear phase," *IEEE Trans. Circuit Theory*, vol. 19, no. 2, pp. 189–194, Mar. 1972.

[101] M. Morelli and U. Mengali, "Feedforward frequency estimation for PSK: a tutorial review," *Transactions on Emerging Telecommunications Technologies*, vol. 9, no. 2, pp. 103–116, Mar./Apr. 1998.

[102] Y. Liao, "Phase and frequency estimation: High-accuracy and low-complexity techniques," Master's thesis, Worcester Polytechnic Institute, Worcester, MA, May 2011.

[103] C. Monti, A. Saitto, and D. Valletta, "Indoor radio channel models for IEEE 802.15.4 technology," Telespazio S.p.A, Tech. Rep., Jun. 2008.

[104] (2017) Fading channels. MathWorks. Natick, MA. [Online]. Available: www.mathworks.com/help/comm/ug/fading-channels.html

[105] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015.

[106] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *International Conference on Learning Representations*, 2017.

[107] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *International Conference on Machine Learning*, Stockholm, Sweden, Jul. 2018.

[108] Y. Shi, K. Davaslioglu, and Y. E. Sagduyu, "Generative adversarial network for wireless signal spoofing," *arXiv:1905.01008*, May 2019.

[109] A. Antoniou, A. Storkey, and H. Edwards, "Data augmentation generative adversarial networks," *arXiv:1711.04340v3*, Mar. 2018.

[110] I. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," *arXiv:1701.00160v4*, 2017.

[111] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, Atlanta, GA, Jun. 2013.

[112] S. U. Rehman, K. W. Sowerby, and C. Coghill, "Analysis of receiver front end on the performance of RF fingerprinting," in *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC)*, Sep. 2012, pp. 2494–2499.

[113] S. U. Rehman, K. W. Sowerby, and C. Coghill, "Analysis of impersonation attacks on systems using RF fingerprinting and low-end receivers," *Journal of Computer and System Sciences*, vol. 80, no. 3, pp. 591–601, May 2014.

[114] B. W. Ramsey, T. D. Stubbs, B. E. Mullins, M. A. Temple, and M. A. Buckner, "Wireless infrastructure protection using low-cost radio frequency fingerprinting receivers," *International Journal of Critical Infrastructure Protection*, vol. 8, pp. 27–39, Jan. 2015.

[115] S. U. Rehman, K. W. Sowerby, S. Alam, and I. Ardekani, "Portability of an RF fingerprint of a wireless transmitter," in *2014 IEEE Conference on Communications and Network Security*, San Francisco, CA, Oct. 2014, pp. 151–156.

[116] Ö. H. Tekbaş, N. Serinken, and O. Üreten, "An experimental performance evaluation of a novel radio-transmitter identification system under diverse environmental conditions," *Canadian Journal of Electrical and Computer Engineering*, vol. 29, no. 3, pp. 203–209, Jul. 2004.